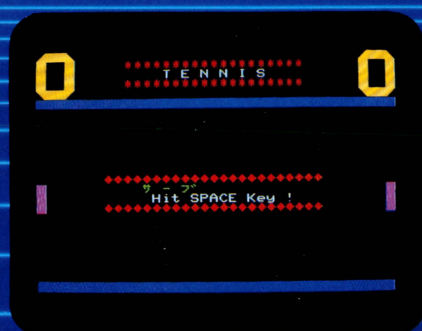
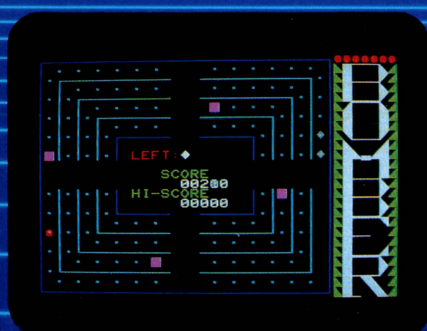
GAMING

GAMINGへの招待

MULTIマイコン研究会　塚越 一雄　著

スペースウォー，テニス，ボンバー，インベーダー

4ジャンルのゲームプログラムのノウハウを全公開

BASICからマシン語までゲームプログラム入門!

たしかな技術で世界をむすぶ

NEC

登場!
クラスを超えた
ハイ・クォリティ・パソコン
パソコン界の先頭を走るNECが、今また話題を独占します。数々の新機能を身につけた高性能マシンPC-6001MKII。

武田鉄矢

画面の絵はハメコミ合成です。

デジタル人にうれしいニュース。

PC-60M43(M)

PC-60M75

PC-6001MKII

NECのパソコンファミリー 国内実績No.1

車内でも外出先でも使えるパソコン。
PC-2000 シリーズ
●本体標準価格……………59,800円

新・発・売
大きさA4サイズの本格ビジネスパソコン。
PC-8200 シリーズ
●本体標準価格……………138,000円

パソコンがビジネスをものにした。
PC-8800 シリーズ
●本体標準価格……………228,000円

8ビットのキャリアを活かせる16ビット。
PC-9800 シリーズ
●本体標準価格……………298,000円

強力なOAソフト"LAN"が自慢の16ビット。
N5200 モデル05
●システム標準価格………698,000円
(ディスプレイ、キーボード、フロッピィディスク1台)

進歩1 パソコンが言葉を持った。

ボイスシンセサイザ内蔵で、任意の言葉を声にできます。学習にもプログラミングにも、しゃべるパソコンならコミュニケーションがもっとホットになります。

進歩2 1024文字の漢字が使える。

教育漢字をはじめとする1024文字の漢字ROMを内蔵して、日常語のほとんどをカバーできます。ひらがな表示もできますから、読みやすい日本語感覚でパソコンとやさしくつきあえます。

進歩3 ドット単位の15色グラフィックス。

320×200ドットのカラーグラフィックスではドット単位4色、160×200ドットでは、ドット単位15色までのカラー指定ができます。CRT出力もRGB・RF・ビデオの3系統を装備。鮮やかな色彩がパソコンの世界をいろどります。

進歩4 たっぷり余裕の記憶力。

RAM64KB、ROM最大96KBの大容量メモリを実装。複雑な演算や高度なグラフィック処理もこなして、パソコンの応用範囲を大きく拡げています。

進歩5 テレビ、ビデオ画面へスーパーインポーズ。

本体内にインターフェースを内蔵しました。だからスーパーインポーズユニット(別売)を接続するだけで、テレビ画面とパソコン画面の合成がOK。合成画像のVTR録画も簡単です。

またしても、進歩はNECから。

●**ミュージック・シンセサイザ内蔵**:8オクターブ、三重和音までの音楽演奏が可能です。●**N60m-BASICを標準実装**:マシン語モニタを含むROM32KBで、PC-6000シリーズの数多いソフトウェアの大部分が活用できます。●**最新のゲートアレイ技術**:最先端テクノロジーの採用で信頼性とコストパフォーマンスを大幅にアップ。●**楽しいアプリケーション・ソフト**:その日から使える4種類の本格ソフトがついています。●**ミニフロッピィ・インタフェース内蔵**:データ量が増えても拡張が簡単にできます。●**ボディカラー**:シルバーメタリック(M)とアイボリホワイト(W)の2色があります。

●CPU/メインμPD780C-1(4MHz)サブμPD8049(8MHz)●ROM/BASIC32KB・漢字32KB・CG16KB・音声16KB●RAM/64KB●表示文字例/40文字×20行●表示文字/496種+漢字1024文字●グラフィック機能/320×200ドット(4色)・160×200ドット(15色)・80×40ドット(15色セミグラフィック)●音楽機能/8オクターブ三重和音●音声合成機能内蔵●CMT/2トラック制御ステレオカセットインタフェース●CRT出力/RGB・ビデオ・RF三方式●キーボード/JIS標準規格配列準拠●プリンタインタフェース/パラレルインタフェース内蔵(セントロニクス社仕様準拠)●シリアルインタフェース/PS-232インタフェース(オプション)●FDDインタフェース内蔵(5インチ)●寸法/365(W)×87(H)×260(D)●重量/3.3kg

飛び抜けて新発売 NECパーソナルコンピュータPC-6000シリーズ PC-6001mkII

本体標準価格……………84,800円 14型カラーディスプレイPC—60M43(W),(M)標準価格65,800円/ディスプレイ置台PC—60M75標準価格5,500円

クラスを超えたハイ・クォリティ・パソコン。
PC-6000シリーズ
PC-6001mkII 本体
●本体標準価格……………84,800円

名機、PC-8001の後継パソコン。
PC-8000シリーズ
PC-8001mkII 本体
●本体標準価格……………123,000円

日本電気グループ・NECパソコンインフォメーションセンター
〒108 東京都港区三田3丁目14-10(明治生命三田ビル) ☎03-452-8000(代)

機能敏感、

未来への鮮やかなコミュニケーション・マシンとして高性能を誇る、NECのカラーキャラクター・ディスプレイ。パーソナルコンピュータの多彩な機能を鮮明に、敏感に描き出します。2,000文字を映し出す高解像度ディスプレイ、そして手軽に使える9型グリーンディスプレイなどバラエティ豊かなラインナップ。新しいシャーシの採用で、スリムなフォルムも新登場。幅広い用途にお応えします。

14型カラーディスプレイ
PC-8052 (JC-1401DF)
●入力信号方式/RGB方式●表示文字例/2,000文字 (80文字×25行、5×7ドット) PC-9801、PC-8801、PC-8001MKII用 ………………118,000円

14型モノクロディスプレイ
PC-8851 (JB-1410P2)
●入力信号方式/コンポジット方式●表示文字例/2,000文字 (80文字×25行、5×7ドット) PC-9801、PC-8801用 ……………… 標準価格58,800円

12型カラーディスプレイ
PC-8049N (JC-1206DH)
●入力信号方式/RGB方式●表示文字例/2,000文字 (80文字×25行、5×7ドット): PC-9801、PC-8801、PC-8001MKII用……標準価格158,000円

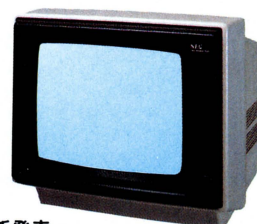
12型カラーディスプレイ
PC-6042K (JC-1202M)
●入力信号方式/コンポジット方式●表示文字例/800文字 (40文字×20行、5×7ドット) PC-6001、PC-6001MKII用 …………標準価格56,800円

12型カラーディスプレイ
PC-8058 (JC-1201DF)
●入力信号方式/RGB方式●表示文字例/2,000文字 (80文字×25行、5×7ドット) PC-9801、PC-8801、PC-8001MKII用 ……標準価格99,800円

日本電気グループNECパソコンインフォメーションセンター 〒108 東京都港区三田三丁目14−10(明治生命三田ビル) ☎(03)452-8000(代)

14型高解像度カラーディスプレイ
PC-8853N (JC-1412P2)
●入力信号方式／RGB方式 ●表示文字例／2,000文字（80文字×25行、5×7ドット）
PC-9801、PC-8801用……標準価格168,000円

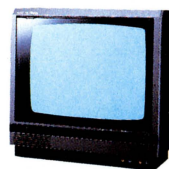
時代直感。

新発売
14型カラーディスプレイ
PC-8054 (JC-1402D)
●入力信号方式／RGB方式 ●表示文字例／2,000文字（80文字×25行、5×7ドット）
PC-9801、PC-8801、PC-8001MKII用………………………………………標準価格65,800円

新発売
12型カラーディスプレイ
PC-8048N (JC-1204D)
●入力信号方式／RGB方式 ●表示文字例／800文字(40文字×20行、5×7ドット)PC-9801、PC-8801、PC-8001MKII用………………………………………標準価格59,800円

新発売
12型高解像度モノクロディスプレイ
PC-8841 (JB-1210P2)
●入力信号方式／コンポジット方式 ●表示文字例／2,000文字(80文字×25行、5×7ドット)
PC-9801、PC-8801用……標準価格44,800円

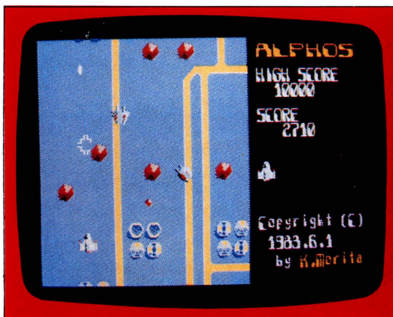
12型グリーンディスプレイ
PC-6041 (JB-1260M)
●入力信号方式/コンポジット方式●表示文字例/512文字 PC-6001使用時(32文字×16行、7×9ドット)、2,000文字(80文字×25行、5×7ドット) PC-9801、PC-8801、PC-8001MKII、PC-6001、PC-6001MKII用………………………標準価格36,800円

12型グリーンディスプレイ
PC-8050N (JB-1205M)
●入力信号方式/コンポジット方式●表示文字例/2,000文字 (80文字×25行、5×7ドット) PC-9801、PC-8801、PC-8001MKII用・標準価格29,800円

9 型グリーンディスプレイ
PC-8046 (JB-902M)
●入力信号方式/コンポジット方式●表示文字例/2,000文字 (80文字×25行、5×7ドット) PC-9801、PC-8801、PC-8001MKII用・標準価格35,800円

14型RGBカラーテレビ
C-14N16PV
●入力信号方式/RGB方式●表示文字例/2,000文字 (80文字×25行、5×7ドット) PC-9801、PC-8801、PC-8001MKII、PC-6001、PC-6001MKII用…………………標準価格125,000円

パソコンライト
PL-5101 (BR) (GY)
C(BR)644-3001・(GY)644-3002
●パソコン専用照明器具 ●15Wレフレクタ・サンホワイト5蛍光ランプ（パソコンライト専用）●グロースタータ式●プラグ付電源コード 180cm………………………………標準価格11,000円

▼全国のマイコンショップで、お求め下さい。▼

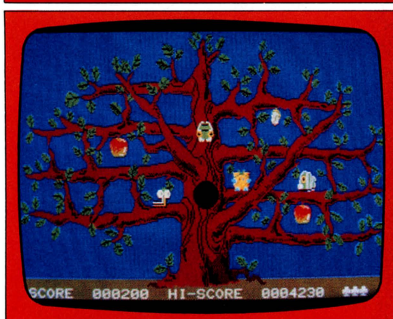
取扱店募集中!

アルフォス

©森田和郎・©株式会社ナムコ

PC-8801・パソピア7

DISK(PC-8801用)・6,800円

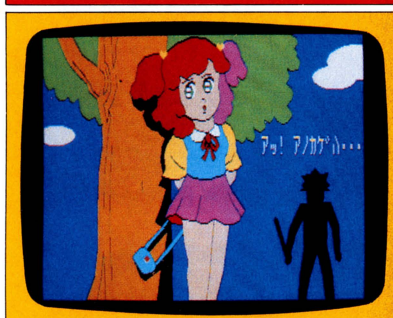
★作者　森田和郎★――――4,800円

君は今高度1000フィートの大空を飛翔する。
次々と目の前に謎の飛行物体が迫り、地上からは激しい対空砲火。君は撃って撃って撃ちまくれ!めざすは誰も見た者がいないあの巨大なるアルフォスを破壊することだ。
パソコンの限界を超越したスクロールゲーム

ニュートロン

PC-8801

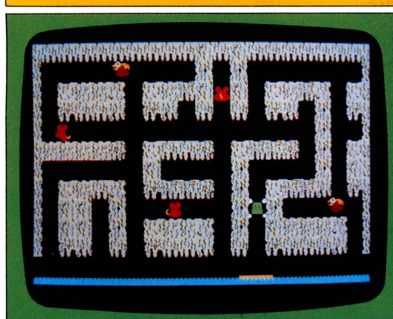
DISK(PC-8801用)…5,800円

★作者　中村光一★――――3,800円

君はロン君。ニュートン村のフルーツの木に黄金の木の実を採りに来たんだけど……その木にはいやな虫たちがいっぱい。僕はヘトヘト。どうしちゃおう?ドア・ドアをしのぐ、おもしろドキドキのファンタジーゲーム。

女子寮パニック

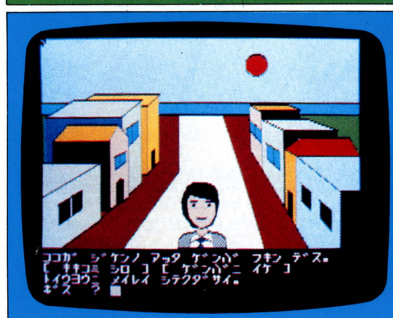
PC-8801
FM-7・FM-8

★作者　槇村ただし★――――3,800円

主人公ロムちゃんは女学生。君は彼女に一目惚れ。この胸の内をロムちゃんに伝えたく、深夜、女の園へもぐり込んじゃおう!ハラハラドキドキ、ハプニングの連続。ロムちゃんと君の恋はどうなるの!?槇村氏のユニークなアイデアを盛り込んだ愉快な学園アドベンチャー

ライト・フリッパー

PC-8801

★作者　岡田良行★――――3,600円

あなたは誤って鐘乳洞へ落ちてしまった。何が何でも生き延びなくてはならない。出口をめざし登り続けるあなたの道筋には数々の危険が!その危険を乗り越えないとあなたはオダブツ…下を流れる地下水へドブン!
アイデア盛りだくさん、スリルと興奮のリアルタイムゲーム。

ポートピア連続殺人事件

PC-8801・8001mkII
PC-6001(32K)・mkII・FM-7/8

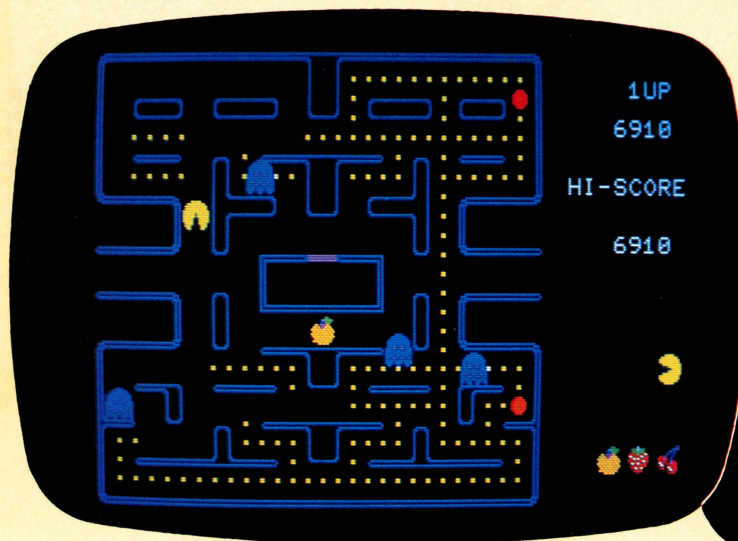
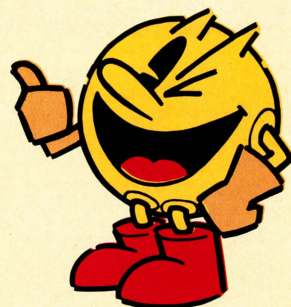
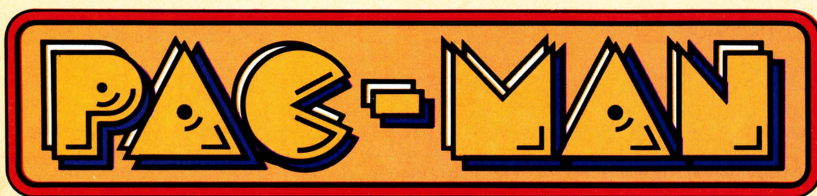
★作者　堀井雄二★――――3,600円

ある夜おこった密室殺人。キミは部下のヤスをひきつれ捜査にのりだすが、次々死んでゆく容疑者たち。
犯人は誰か!港神戸を発端に、舞台は京都から淡路島へ……。
「君は犯人を追いつめることができるだろうかっ?」サスペンスアドベンチャーの決定版!

マイコンソフト

namcoオリジナ

ゲーム・センターの大ヒット作品「パックマン」，「ギャラクシアン」，「ディグダグ」をはじめ，最近では「ゼビウス」などを制作している(株)ナムコと，電波新聞社が協力して開発した，パソコン用ソフト・テープが発売になりました。

今月ご紹介するのはその第一弾で，「namcoオリジナルゲーム・シリーズ」と題され，おなじみのゲームが，本物そのままの迫力で楽しめます。もちろんゲーム・マシーンとパソコンとの間には，ハード上の差がありますが，いかにオリジナルにせまれるか，越えられるかを最大のポイントとして開発した自信作です。ぜひお楽しみください。

◀シャープX1用パックマンの画面。スピード，サウンド，動き共にリアル！キミは何面までクリアできるかな？

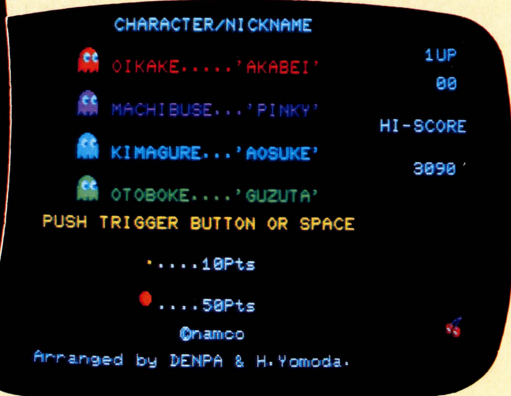
▲スタート画面。もちろんミュージックもでるゾ！

アーケード・ゲームにチャレンジ！

ゲーム・プログラム大募集！

電波新聞社は，有力ゲーム・メーカーの(株)ナムコ，(株)タイトー，両社と契約を結び，ビデオ・ゲームをパソコン用に映像複製することができます。

みなさんのプログラミング能力を生かした，実物に負けない内容のパソコン用プログラムがあれば，ぜひお知らせください。優秀作品は「オリジナルゲーム・シリーズ」に加え，発売させていただきます。

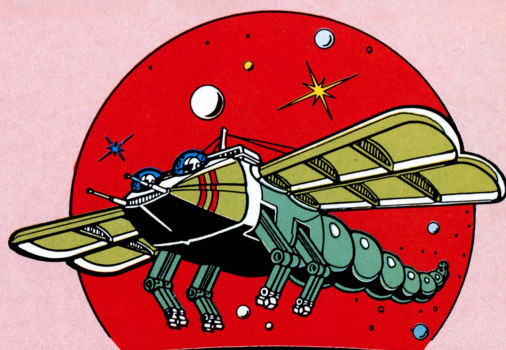
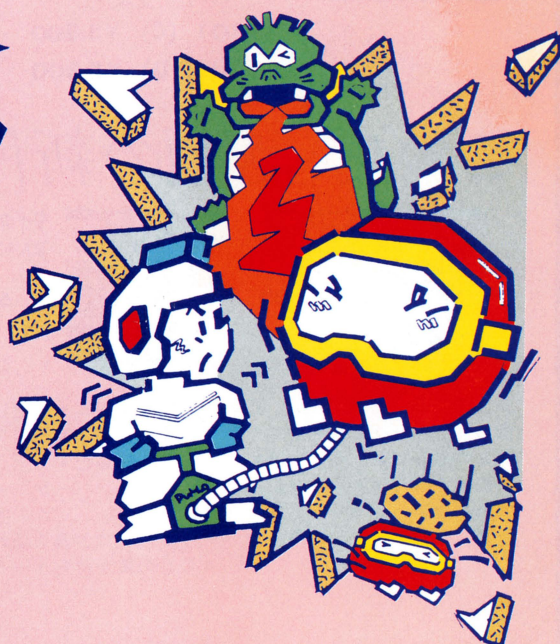
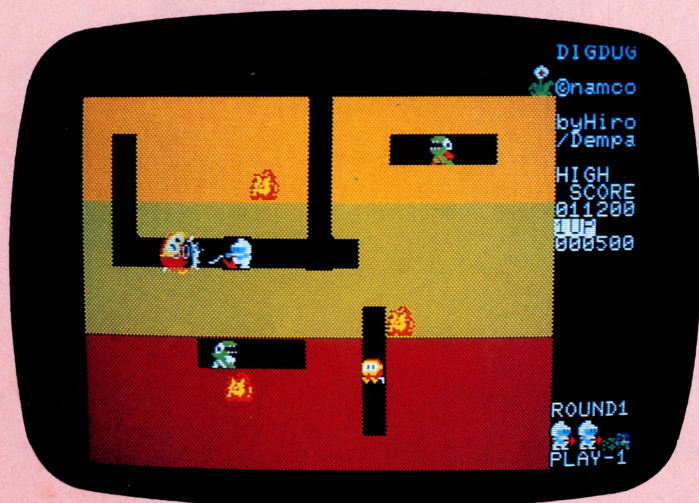
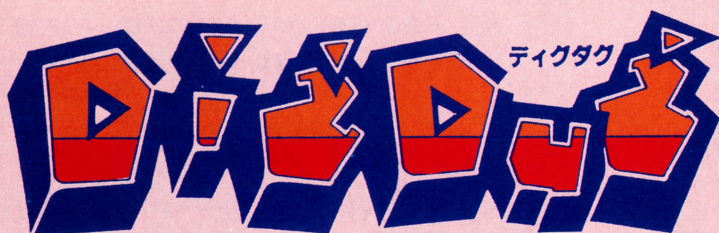
※プログラム商品化に際しては，著作権使用料をお支払いいたします。

◎namcoオリジナルゲーム・シリーズの内容◎

題　名	対応機種	定　価	
パックマン	FM7	3,500円	発売中
	X1	3,500円	
ギャラクシアン	MZ-700	3,000円	発売中
ディグダグ	PC-8001/8801(N)	3,000円	近日発売
	X1	3,500円	近日発売

※対応機種は続々と増加いたします。
※別タイトルのゲームも発売になります。
※すべて美しいデザインのビニール・レザー・ケースにおさめられております。
※お求めは全国有力マイコンショップ，書店にてどうぞ。

※この件に関するお問合わせは電波新聞社出版部「マイコン・ソフト開発室」☎03-445-6111(代表)にお願いします。

パソコンでゲーム・マシーンに挑戦!!
ル・ゲーム・シリーズ発売!

▲MZ-700の限界にチャレンジ!エイリアンの攻撃は本物なみのはげしさ!キャラクタの工夫も見どころだ!

▲闘志をふるいたたせるスタート画面!

◀X1用ディグダグの画面。ジョイスティックも使えます

どちらも、超！自信作!!

SHADOW PRISON
忍者くん
ADD 90×1
SCORE 000090
HIGH SCORE 000400
LEFT 3

Dream land

忍者くん

とても可愛い忍者くん、どうしても最上階へ行かねばならない。
しかし、七人の恐しい役人が、すきあらばと、忍者くんを狙っているのだ。斬られずに無事、最上階へ到達できるかどうかは、君の腕次第！プレイしている人も、見てるだけの人も、思わず声を上げてしまう程、実におもしろくスリリングなゲームの登場だ。

PC-8801 (C) ¥3,800
PC-8801(FD) ¥5,800

ドリームランド

君は眠りについた——いつのまにか、夢の中とも知らず、さまよい歩く冒険者になり変わっている。様々なグラフィックが織りなす、ファンタジックで奇妙な、不思議な世界にいる。君の冒険心はさらにかきたてられるのだ。この幻想の夢から、果たしていつ覚めるのか？　かつてないスケールのビッグアドベンチャーゲーム。画面イラストも60を超える超大作。君のアドベンチャースピリットにロマンが加わる。

新発売 MZ-80B MZ-2000(FD)—2枚組
開発中 PC-8801 PC-9801(FD)
¥12,800

この夏、話題の新作ソフトずら〜り勢ぞろい!!

ピコピコ

MZ-2000
X1
¥3,800(C)

36個のブロックを、攻撃、防御に使ったり、自ら発射する弾丸で敵を攻撃。死体は消えずに残るのでゲームが進むにつれ、複雑化してくる。敵はたまにワープするので御用心。

ダイヤモンドアドベンチャー

PC-6001
X1
¥3,800(C)

三階建てのビルに隠された、時価10,000,000円のダイヤモンドを探し出す。デジタルロックのナンバーがキーポイント！しかし、ビルの中はトリックだらけ、君を陥れるいくつもの罠が待ちうけています。

ミッドナイトコマンダー

PC-8801
FM-7
¥3,800(C)

真夜中の洋上、月あかりもなく、レーダーを頼りに敵を探索し、攻撃する。ちょっとしたミスから敵のレーダー網にかかり攻撃を受けたらたちどころに撃沈されてしまう。味方の艦隊、飛行機隊を駆使し、敵の本拠地を乗っ取れ！

エキサイティングゴルフ

FM-7
¥3,500(C)

ゴルフは、もう芝の上でするだけのものではなくなった!!グラフィックに展開されるコースは、もう君だけのもの！君の緻密な計算から打たれたボールはグリーンを捕らえることができるか？臨場感あふれる本格的なゴルフシミュレーションゲーム。

マイクロキャビン

販売／㈱ MICRO CABIN
企画・制作／㈲ ARROWSOFT

〒510 三重県四日市市鵜の森1-2-15 メゾンヴァンベール2F ☎0593-51-6482

まえがき

マイコン・ホビーにおける最もエキサイティングな分野――それが

マイコン・ゲーム

です。画面狭しと動きまわるグラフィックのスター達，一瞬一瞬が，勝負の**リアルタイム・ゲーム**。また何時間も費やし，静かな中にも的確な作戦を要求される**シミュレーション・ゲーム**。マイコンとの熾烈な知恵比べがくりひろげられる**思考ゲーム**。またみんなでガヤガヤ，**マイコン・パズル**――。沢山の種類の沢山のゲームが，プログラム一つで変化して行きます。

あなたもマイコン・ゲームを作ってみたいと思いませんか？　あなた自身のオリジナルなゲームを。ゲームのプログラム――，そこにはありとあらゆるソフト・テクニックが展開されて行きます。ゲームを作ることで，あなたのプログラム・テクニックが格段に向上して行くことでしょう。

「GAMING への招待」は，もともとは月刊「マイコン誌」の「入門コーナー」における一つの連載でした。それは81年12月号から始まり，今でも続いています。その内容は，

各自のレベルを問わず

ゲーム作りを楽しもう！

とするもので，一つのＧＡＭＥを何回かに分け，毎回少しずつ作っていくのを基本パターンにしています。

各シリーズは，完全に独立しています。どこから読み始められても結構です。本書にはその中から四つのシリーズが集められています。そして今後も第２巻，第３巻とコレクションが増えて行くでしょう。

あらゆるレベルの

あらゆるジャンルのゲーム

が集められている。また，

あらゆるソフト・テクニックの集大成

それが，**「ＧＡＭＩＮＧへの招待」**です。ごゆっくりとお楽しみください。そしてあなた自身のオリジナル・ゲームを作ってください。それがフィード・バックされてくるのを楽しみに待っています。

ＭＵＬＴＩマイコン研究会

塚　越　一　雄

もくじ

第1ブロック 「SPACE WAR」に挑戦

第1章 画面作りに挑戦

第2章 SPACE WAR変身？

第2ブロック 元祖「テニス・ゲーム」に挑戦

第1章 オリエンテーション

第2章 図形を動かす

第3章 キー・スキャン

第4章 テニスゲーム完成！

第5章 マシン語による高速化

第3ブロック リアルタイムゲーム「BOMBER」に挑戦

第1章 オリエンテーション

第2章 GAME仕様の分析

第3章 SKIPマークの導入

第4章 キー入力について

第5章 REDCARの追撃

第6章 仮の最終回

第4ブロック「インベーダー・パニック」に挑戦

第 ブロック

SPACE WARに挑戦

20世紀フォックス「スター・ウォーズ/ジェダイの復讐」より

画面作りに挑戦

GAME作り——否，プログラミング一般を非常に難しいものだと思っている人がいます。本当でしょうか？

この問に対する答——難しいですね。プログラミングって奥の深いものだし，やればやる程その難しさがわかってくる。また易しいと言えば易しい。

これでは，答になりませんね。ただ一つ，次のことだけは言えると思います。

ＢＡＳＩＣを例にとります。プロミラミングの出来ない（と本人は思い込んでいる）人は，

ＢＡＳＩＣ言語の全て

をマスターしないとプログラムは作れないものと思っているようです。たとえば次のＩ氏のように。

「マイコンを買って約１年になります。仕事は家が鋳物関係をやっているので，それを手伝っています。マイコンを買ったのは家の会計処理をやらせようど思ったから。それでＢＡＳＩＣの本を買って読んでいるんですが，難しいですね。初めのＰＲＩＮＴのあたりはわかっていたんですが，変数やＩＦが出てきたらどうも良くわからなくなってきた。だからまだ会計処理は出来ていません。ＥＲＡＳＥって，どうやって使うんですか？」

Ｉ氏は，マイコン歴１年。最初は本体とグリーン・ディスプレイを購入。そのうちプリンタがあった方が良さそうだからと言ってドット・インパクト・プリンタを購入。その後ＤＩＳＫも購入しています。しかし，まだ自作のプログラムは作っていません。プログラミング教本は，何度も眺めていますから

難しい命令の名前

だけはいろいろと知っています。

もう１人紹介しましょう。

Ｋ君。高校二年生です。彼はやたらにマシンをいじくります。自分のもさることながら，人のや，マイコンショップのものまで。自分でマシンを手に入れると凄いもので，マニュアルを

１行読んでは，その使い方を実験

してみる。マニュアルに書いてないことまで発見してしまう。果ては，マニュアルの間違いを発見してしまうほどです。そして，一つ命令を覚えるとその命令の応用を考えてみる。否，実際にキー・インして確かめる。

“ア，こんな使い方があった！

ア，こんなことにも使える！”

発見の連続です。

彼はヒマがあると，**自分の今までの知識内で作れるプログラムを開発**しています。そして，３か月もしないうちにＴＶゲームを作り上げてしまいました。それでも彼の知っている命令は，マニュアルの前の方三分の一くらいでしょう。

そこで先の問に関する解答です。

自分の知っている命令だけでプログラムを作れば，プログラミングなんてやさしいですよ。エッ？　ＰＲＩＮＴ文しか知らない。結構，ＰＲＩＮＴ文だけで出来ることをやれば良いのです。もし不合理な使い方をすれば，あなたのマシンが**エラー・メッセージ**で警告してくれます。いわば，

あなたのマシンがあなたのプログラミングの先生
です。ものは，ためし。次節で
ＰＲＩＮＴだけで出来ること
をやってみましょう。

スペース・ゲームに挑戦

まずコンピュータを１台，御用意ください。以下に実習の様子を示しましょう。

――コンピュータを用意しろっと言ったって，コンピュータ持っていないんじゃない！
――いや，コンピュータと言っても，自分のでいいのさ，ホラ，それ。
――何だ，自分のマイコンじゃないか。
――マイコンだって立派なコンピュータさ。
――僕，今マイコン持ってないから，一緒にやらせてよ。
――ムサクルシイなあ。
――“まず，画面をクリアして”と言ってるよ。このテレビ，もうクリアされているね。
――馬鹿，まだスイッチ入れていない。
(効果音：パチン。間――)
――あっ，ついてきた。Ｂ・Ａ・Ｓ・Ｉ・Ｃ，Ｖ・Ｅ・Ｒ・Ｓ・Ｉ・Ｏ・Ｎ？何だ，コリャ？
――ここを消せば，いいんだね。横棒を押していくと。
――コラ，コラ！スペースで画面を消している。かわいそうだね。この子は。このマシンは，ＣＬＲキーを押せば一発じゃ！。
――アッ！　消えた。天才！

以上のように御自分のマイコンを用意し，電源をＯＮにし，ＢＡＳＩＣが使える状態にしてください。もしあなたのマシンが，画面モードの切り換え可能なものでしたら，お好きなモードにセットしておいてください。ただし，これからＴＶ画面に絵を描きますから，
行間にスキマのあかないモード
を選択してください。私のマシンは，ＰＣ－8001ですから１画面20行モードにすると，絵と絵の間にスキマがあいてしまいます。
私の場合，

　　ＷＩＤＴＨ　40，25
　　ＣＯＮＳＯＬＥ　0，25，0，0

にセットしてみました。これは，

　　画面サイズ＝40×25
　　白黒モード

その他の設定を行っています。

以上で絵を作る準備はできました。まず画面をクリアします。これは，マシンによってそれぞれやり方が異なりますね。**ワン・キー一発**のものがあれば，**コマンド処理**の必要なものまで千差万別ですね。各自のマニュアルにしたがってください。

そして次が，もっとも楽しい時間です。カーソル移動キーを使ってカーソルを画面の好きな位置にもっていってください。そしてキー・ボードから好きなキャラクタを打ち込むのです。もしあなたのマシンがグラフィック・パターンの使えるものでしたら，沢山使ってみてください。私のＰＣ－8001も，もちろん使えます。

こうして，ひたすら１時間。おもいっきり童心にかえって，**ＴＶ画面に好きな絵を作ってみてください。**

　　どんなキャラクタを組み合わせるか
　　どんな画面を作るか

ここらあたり，センスの違いが出てくるところです。

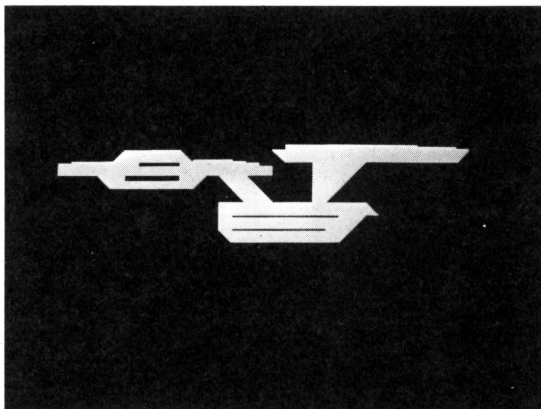
絵を作るにあたって，二，三注意しておきます。

①　絵は，画面一ぱいに作るのではなく，
まわりを少しあけて
おいてください。その方があとの作業をやりやすくなります。

②　出来るだけ手を抜かず，**消すのがもったいないくらいのもの**を描いてください。

③　最近の大低のマシンは，スクリーン・エディタ機能がついています。ですからまず**大まかな画面**を作り，あとは自分で気に入るまで何度も何度も**細かく修正**してみてください。

さあ，出来ましたでしょうか？

今回私は，スペース・ゲームの宇宙船に挑戦してみ

《写真１》これが宇宙船？

ました。**写真1**がそれです。笑ってやってください。みっともないですね。あなたは，何を描いてみましたか？　きっとあなたのディスプレイ上に，素晴しい画面ができあがっていることと思います。どうですか？消すのがもったいなくなった！　のではないでしょうか？

？文の登場――ハテナ？

――出来た。出来た。芸術的だなあ。
――退廃的だなあ。
――僕が手を加えたからずっと見栄えが良くなった。
――お前がよけい手を加えたから，ひんまがってしまった。
――ア～～，これ消すのもったいないなァ。
――早く電源を落とそう。

ここで電源を切ってしまうと，今せっかく作った作品が一瞬のうちに消え去ってしまいます。そこで登場するのは，あなたの良く知っている

PRINT文

です。

それでは，具体的な作業に入りましょう。なお最初にくれぐれも御注意申し上げておきますが，以下の作業は，ぜひ**慎重に**行ってください。さもないと，たいせつなあなたの絵をダメにしてしまいます。たとえば，もしあなたのマシンに〝画面消去用のボタン〟（PC－8001では，CLRキーです）がついているなら，ぜったいにそのキーに手を触れてはいけません。

さあ，作業を開始しましょう。まずカーソル移動キーで，カーソルを絵の描かれている行の左端にもって行きます。そうしたらおもむろに

10　？　〝

とキーインしてください。次にカーソルを，絵の右端に持って行きます。そして，〝をキーインします。するとその行は，

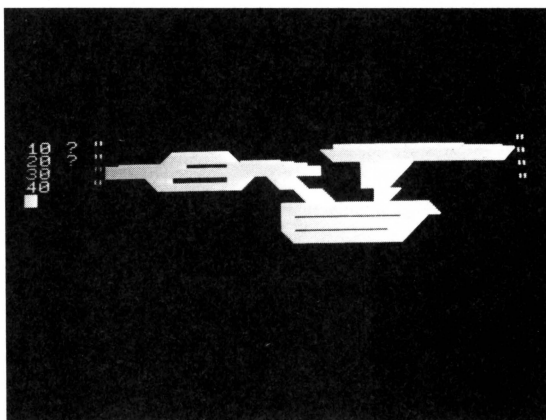
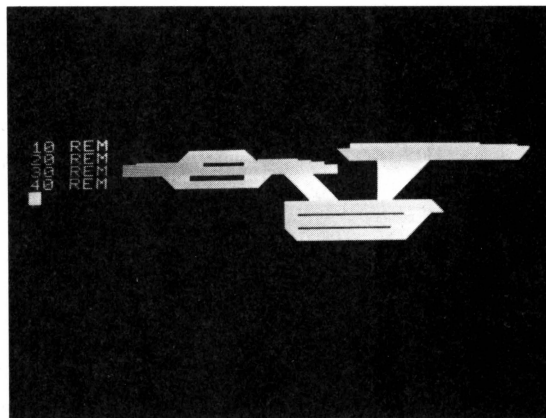
10　？　〝〈1行目の絵〉〟　――①

という構成になります。そうしたら改行キー（PC－8001ならRETキー）を押してください。

どうですか？　もう何をやろうとしているのかおわかりですね？　①は，BASICのPRINT文を入力しているのと同じです。すなわち今あなたが作った画面を，あなたのよく知っているPRINT文を用いて

BASICのプログラムにしてしまう

のです。

㊟　？がPRINTの代わりになるのは，御存知ですね。この省略形が使えないマシンでは，PRINTとキーインしてください。また，もしそのスペースが

《写真2》？(PRINT文)で絵ととり込む

《写真3》REN文で絵をとり込む

《リスト1-1》PRINT文だけで絵をプログラム化

```
1100 PRINT"                                  "
1110 PRINT"                                  "
1120 PRINT"                                  "
1130 PRINT"                                  "
1140 PRINT"                                  "
1150 PRINT"                                  "
1160 PRINT"                                  "
1170 PRINT"                                  "
1180 PRINT"                                  "
```

ないときは，とにかくBASICのプログラムに組み入れてしまえば良いのですからREMの注釈文か，DATA文にしてしまえば良いでしょう。あとでもう一度スクリーン・エディタでPRINT文になおせばよいのです。

以下，次々とこの作業を繰り返します。**写真2**は，4行目まで作業が終り，カーソルが，5行目の先頭に移動しています。また**写真3**は，同じことをREM文を用いて実行したところです。

こうしてすべての絵をPRINT文に組み込んでしまえば，あとはいくら絵をこわしても大丈夫です。**リスト1-1**が出来あがったプログラムです。

手順と重要な注意

以上の手順，おわかりいただけたでしょうか？

あなたは，BASIC言語のPRINT文しか知りません。でも，

スペース・ゲーム
宇　宙　船表示のプログラムが作れたのです！

以上の手順，重要ですからもう一度まとめておきましょう。

① マシンの電源をONにする。
② 画面をクリアする。
③ 画面の中央部に表示したい絵を書く。
④ スクリーン・エディタを使って
　　　行番号　？〝　〃
　を追加する
⑤ これでおしまい。

どうですか？　簡単ですね？　でも簡単なわりには，結構ものすごいことができるでしょう？

ところであなたの作った絵，たしかにプログラムに保存されました。でもまだ安心してはいけません。もしかしたら

停　電　！

が起こるかもしれません。するとせっかくのプログラムも一瞬のうちに消えてしまいます。さあ，早くカセットに録音しておきましょう。クワバラ，クワバラ。

ところでこのこと，馬鹿にしてはいけません。プログラムは製作進行と同時に

細かく細かくSAVEを繰り返す

ことが重要です。たとえば，休日，1日をかけて長いプログラムを作ったとします。そして夕方，ようやく完成したところで初めてプログラムをSAVEしようとします。そのときCSAVE（PCのセーブ・コマンドです）とするところを，うっかり間違えて，CLOAD（PCのロード・コマンドです）とキーインしてしまったとします。あるいは，うっかり電源コードをひっかけてしまったとします。**アーメン〜〜〜！プログラムは一瞬のうちに消えてしまいます。**こんなときは，いくら悔やんでもプログラムは戻ってきません。そして，おそらく二度と同じものを作る気は起こらないでしょう。

こんなことは，私もそうですし，多くの人が経験していることでしょう。もし，細かく，細かくプログラムをセーブしておけば，

被害は最少限に済ますことができた

ことでしょう。

以上のように，将来あなたが長いプログラムを作るようになったとき，プログラムのセーブはこまめに行うことをお勧めします。そんなときのために，今のうちからその習慣をつけておきましょう。さらに言えば，プログラム，データ類は

定期的に**バック・アップ**を取る

のが良いと思います。それもできれば異なる媒体で。なぜなら，たとえばDISKがこわれて修理屋行きとなったとしても，その間カセットで代用できるわけです。

〈教　訓〉
プログラムのバック・アップは「こまめ」に！

プログラムの整形

さあ，SAVEは済みましたか？　完全に安全な状態になったところで，出来上がったプログラムを走らせてみましょう。私の場合でしたら，**リスト1-1**のプログラムを走らせるわけです。

RUN↘

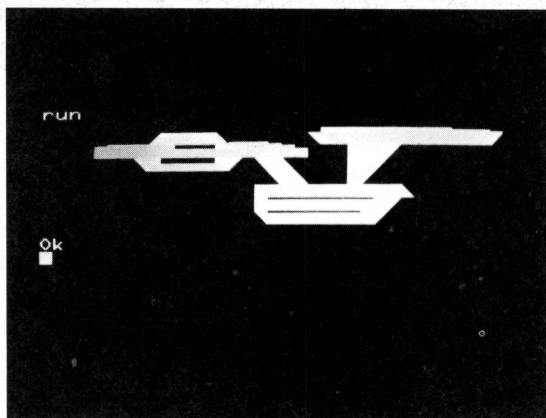
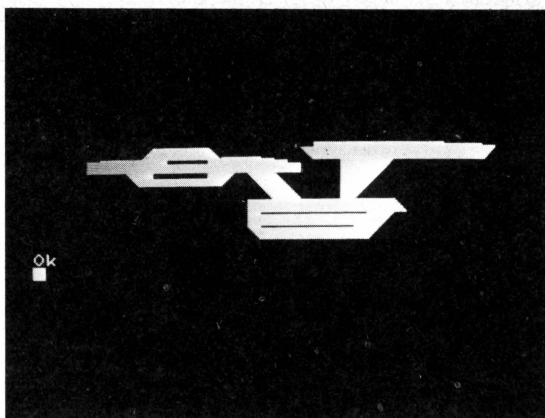
OK？　**写真4**が走らせたところです。

どうですか？　どうも気に入りませんね。

前の画面が残ってしまう！
うまく真中に表示されない！

最初に作った絵は，ちゃんと真中に書いたはずなのに。

プログラムというものは大体一回で気に入ったものが出来るということはまずありません。せっかく作ったプログラムです。さっそく改訂作業に入りましょう。

プログラムを走らせて，**前の画面が残ってしまう**と

《写真４》リスト1-1の実行画面

《写真５》リスト1-2の実行画面

《リスト1-2》REM文を覚えよう

```
1000 REM ====================================
1010 REM   PRINT SPACE WAR ( for LIST2 )
1020 REM          82.4.4:by K. TSUKAGOSHI
1030 REM ====================================
1040 REM
1050 REM ----- PRINT PICTURE ----------------------------------
1060 PRINT CHR$(12);                         ' ｶﾞﾒﾝ ｸﾘｱ
1070 PRINT:PRINT                             ' 2 ｷﾞｮｳ ｶｲｷﾞｮｳ
1080 PRINT"
1090 PRINT"
1100 PRINT"
1110 PRINT"
1120 PRINT"
1130 PRINT"
1140 PRINT"
1150 PRINT"
1160 PRINT"
```

いうのは，常についてくる問題です。これを解決するには，プログラムの最初のところに**画面クリア**の命令を入れておく必要があります。あなたのマシンのマニュアルで

〝画面クリア〞の命令

を探してください。ＰＣ−8001では，ＰＲＩＮＴ　ＣＨＲ＄（12）；というのがそれにあたります。

次にこのままでは，画面の１番上から表示されてしまいます。上部を２〜３行あけたいですね。改行するのもＰＲＩＮＴ文で出来ます。単にＰＲＩＮＴとすれば何も表示されませんから，改行されます。当り前ですね。もし２行改行したければ，ＰＲＩＮＴ：ＰＲＩＮＴとマルチ・ステートメントにすれば良いのです。

最後にせっかく作ったプログラムです。

プログラムのタイトル，製作年月日

くらいは残しておきましょう。せっかくＰＲＩＮＴ文を突破したのです。ついでですから，**ＲＥＭ文**も覚えてプログラムの注釈をつけておきましょう。

リスト1-2が，私の出来上がったプログラムです。また**写真５**が，それを走らせたものです。今度は真中に表示されましたね。

燃える情熱——次の目標は

——スペース・ゲームの絵，出来たか？

——出けた。出けた。

——ハヨ，走らせてみいな！

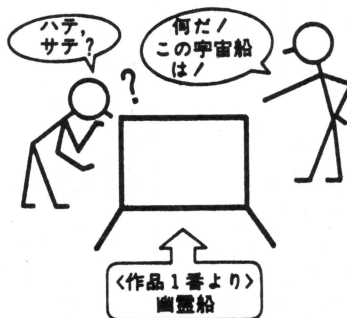
——ヨカ。ホイ。（ら・ん，ポンと）

——何だこの宇　宙　船は！　下の方が欠けてしまっているではないか！

——ハテ，サテ？

——ＰＲＩＮＴ文にする時，失敗して消してしまったな？　下の方が取れてしまっておる！

——幽霊船です。

ＰＲＩＮＴ文によるプログラム作り，お気にめしましたか？　もっとやってみたい？　結構ですね。せっかく宇　宙　船を作ったのです。せっかくですから，もう少し**飾り**をつけて，もっともっとＧＡＭＥらしくしてみましょう。

第2章 SPACE WAR変身？

もっとGAMEらしく

写真5をもう一度見てみましょう。宇宙船が写っています。これに何を追加しましょうか？

タイトル？

そうですね。タイトルを追加しましょう。手順は，次のようになります。

① **リスト1－2**のプログラム（またはあなたの作られたプログラム）を，マシンにロードします。

② これからこのプログラムにスクリーン・エディタを使って絵を追加します。もしプログラムを前の方に追加するなら，**リナンバーをかけて**行番号の始めの方をあけておくと良いでしょう。

③ 画面をクリアする。以下，手順は先と同じになります。

④ 絵を書く。

⑤ スクリーン・エディタで絵をＰＲＩＮＴ文に取り込む。このとき，行番号が最初のプログラムと重ならないように注意します。

写真6をご覧ください。これが私の描いた“ＳＰＡＣＥ　ＷＡＲ”のタイトルです。続いて**写真7**で，スクリーン・エディタでプログラムに取り込んでいます。こうして出来上がったプログラムが，**リスト1－3**です。

カラー版の完成！

リスト1－3を簡単に説明しておきます。

① プログラムは，行番号1000から10刻みにリナンバーがかけてあります。

② **1000～1050**
注釈で，プログラムのタイトルと製作年月日等を記録してあります。

③ **1060**
これはＰＣ独自のハードに関する部分です。

④ **1070**
画面クリアです。

⑤ **1080～1120**
今作ったタイトルの部分です。

《リスト1-3》SPACE　WARタイトル完成間近

```
1000 REM ================================
1010 REM    PRINT SPACE WAR ( for LIST3 )
1020 REM            82.5.6:by K. TSUKAGOSHI
1030 REM ================================
1040 REM
1050 REM ----- PRINT PICTURE ----------------------------------
1060 WIDTH 40,25:CONSOLE 0,25,0,0                   ' ｶﾞﾒﾝ ｾｯﾃｲ
1070 PRINT CHR$(12);                                ' ｶﾞﾒﾝ ｸﾘｱ
1080 PRINT"                                "
1090 PRINT"                                "
1100 PRINT"                                "
1110 PRINT"                                "
1120 PRINT"                                "
1130 PRINT:PRINT                                    ' 2 ｷﾞｮｳ ｶｲｷﾞｮｳ
1140 PRINT"                                "
1150 PRINT"                                "
1160 PRINT"                                "
1170 PRINT"                                "
1180 PRINT"                                "
1190 PRINT"                                "
1200 PRINT"                                "
1210 PRINT"                                "
1220 PRINT"                                "
```

《写真 6》まずタイトルをデザインする

《写真 7》PRINT文でプログラム化

⑥　1130

プログラムを走らせた結果，タイトルと宇宙船の間にすき間をあけた方が良いことがわかったので，2行改行してみました。

⑦　1140〜1220

最初に作った宇宙船の絵です。

写真8が**リスト1－3**を走らせたものです。だいぶGAMEのタイトルらしくなってきたでしょう。

この要領で，どんどんプログラムを改訂していってください。あとは，あなたのアイデア次第です。もしあなたのマシンでカラーが使えるなら，カラー化すると良いでしょう。

私も，私のマシンで**カラー化**してみました。プログラム例を2本作ってみましたので，御紹介しておきます。**リスト1－4**，**リスト1－5**が完成したプログラム。そして**写真9**，**写真10**が走らせたところです。マアマア，GAMEらしくなったでしょう？　ここで注目してもらいたいことは，**リスト1－4**，**リスト1－5**において色をつける部分等を除いて，

その本体はすべてPRINT文で出来ている.

ということです。PRINT文を使うだけでも，これくらいのことは出来るのですね。

まとめ

我々は，PRINT文を駆使して“SPACE WAR”に挑戦してきました。そろそろ，まとめに入り

《リスト1-4》カラー版SPACE WAR I

```
1000 REM ================================
1010 REM    PRINT SPACE WAR ( for LIST4 )
1020 REM          82.5.6:by K. TSUKAGOSHI
1030 REM ================================
1040 REM
1050 REM ----- SET TV MODE ---------------------------------
1060 WIDTH 40,25:CONSOLE 0,25,0,1                ' ガメン セッテイ:カラー モード
1070 PRINT CHR$(12);                             ' ガメン クリア
1080 REM
1090 REM ----- PRINT TITLE ---------------------------------
1100 COLOR 2                                     ' アカ
1110 PRINT"####################################"
1120 COLOR 3                                     ' パープル
1130 PRINT"                                    "
1140 COLOR 6                                     ' キイロ
1150 PRINT"                                    "
1160 PRINT"                                    "
1170 PRINT"                                    "
1180 PRINT"                                    "
1190 PRINT"                                    "
1200 PRINT
1210 COLOR 3                                     ' パープル
1220 PRINT"                                    "
1230 COLOR 2                                     ' アカ
1240 PRINT"####################################"
1244 REM
1245 REM ----- PRINT SPACE SHIP ----------------------------
1250 COLOR 5                                     ' シアン
1260 PRINT:PRINT                                 ' 2 ギョウ カイギョウ
1270 PRINT"                                    "
1280 PRINT"                                    "
1290 PRINT"                                    "
1300 PRINT"                                    "
1310 PRINT"                                    "
1320 COLOR 1                                     ' ブルー
1330 PRINT"                                    "
1340 PRINT"                                    "
1350 PRINT"                                    "
1360 PRINT"                                    "
1364 REM
1365 REM ----- PRINT SUB TITLE -----------------------------
1370 PRINT
1380 COLOR 4                                     ' シアン
1390 PRINT"          ============================"
1400 COLOR 7                                     ' シロ
1410 PRINT"            1982.5.3                 "
1420 PRINT"               PRODUCED BY K.TSUKAGOSHI "
1430 COLOR 4                                     ' シアン
1440 PRINT"          ============================";
1450 GOTO 1450                                   ' ループ
```

《写真８》リスト1-3の実行画面

《写真９》リスト1-4の実行画面

ましょう。

本ブロックの最初の方で，私は高校生Ｋ君を御紹介しました。彼は，即実行派です。マニュアルを１行読んでは，すぐに実験で確めます。頭の中で

〝わからない，わからない〟

とはやりません。

〝この命令はこんなことに使えるのか〟

と実験によりわかったことを素直に喜んでいます。だからこそあれだけ短時間のうちに，あれだけのプログラムが作れたのでしょう。

私が，“ＳＰＡＣＥ　ＷＡＲ”の例を通して言いたかったこともここらあたりにあります。消化不良のうちからやたらと新しい命令に手を出すことは，やめましょう。あなたが良く知っている命令を大切にしましょう。それを使うだけでも，プログラムは作れますよ――こう言いたかったわけです。もうあなたは，納得できますね。

そこで，今後のＧＡＭＥ作りをめざして，勉強の方向に目をむけてみましょう。

《リスト1-5》カラー版SPACE　WARⅡ

```
1000 ' ══════════════════════════════
1010 '    PRINT SPACE WAR (ｶﾗｰ )
1020 '           82.4.4:by K. TSUKAGOSHI
1030 ' ══════════════════════════════
1040 '
1050 '----- SET MODE ------------------------------------
1060 WIDTH 40,25:CONSOLE 0,25,0,1              ' 40X25:ｶﾗｰ
1070 PRINT CHR$(12);                           ' ｶﾞﾒﾝ ｸﾘｱ
1080 '
1090 '----- PRINT TITLE ---------------------------------
1100 COLOR 6                                   ' ｷｲﾛ
1110 PRINT"══════════════════════════════════════"
1120 COLOR 2                                   ' ｱｶ
1130 PRINT"                                      "
1140 PRINT"                                      "
1150 PRINT"                                      "
1160 PRINT"                                      "
1170 PRINT"                                      "
1180 PRINT:COLOR 3                             ' ﾑﾗｻｷ
1190 PRINT"-------------------------------------" ' SUB TITLE
1200 COLOR 7                                   ' ｼﾛ
1210 PRINT"                  1982ﾈ 4ｶﾞﾂ 28ﾆﾁ"
1220 PRINT"                    by K. THUKAGOSHI"
1230 COLOR 6                                   ' ｷｲﾛ
1240 PRINT"══════════════════════════════════════"
1250 '
1260 '----- PRINT PICTURE -------------------------------
1270 PRINT:PRINT:COLOR5                        ' ｼｱﾝ
1280 PRINT"                                      "
1290 PRINT"                                      "
1300 PRINT"                                      "
1310 PRINT"                                      "
1320 PRINT"                                      "
1330 PRINT"                                      "
1340 PRINT"                                      "
1350 PRINT"                                      "
1360 PRINT"                                      "
1370 COLOR 4                                   ' ﾐﾄﾞﾘ
1380 PRINT"++++++++++++++++++++++++++++++++++++++++"
```

あなたがマスターしなければならない命令は，おおむね次の二つに分かれます。

① **基本命令系**

ＢＡＳＩＣでしたら，ＰＲＩＮＴを代表に

ＬＥＴ等の代入文

ＩＦ～ＴＨＥＮ等の制御文

等の基本的命令をマスターしてください。難しい命令は不要です。ＧＡＭＥの本体だけなら，これらだけで完全に記述できます。なおこれらの命令のマスターには，**比較的初期の，それも**

ＴＩＮＹ　ＢＡＳＩＣ あたりの参考書

が良いでしょう。そこには，ムダな命令が出てきませんから。

② **ＧＡＭＥを表現する上での特有な命令**

ＧＡＭＥを視覚的に表現し

《写真10》リスト 1-5の実行画面

ようとしたり，リアルタイムGAMEを作ろうとすると，**リアルタイム・キー入力**等の命令と，いくつかの特有なテクニックのマスターが必要です。その数は多くありません。何が必要かは，次の第2ブロック「テニスゲームに挑戦」を御覧ください。ただし，この②の系統は，マシンのハードに依存しますから，

あなたのマシン上でマスター

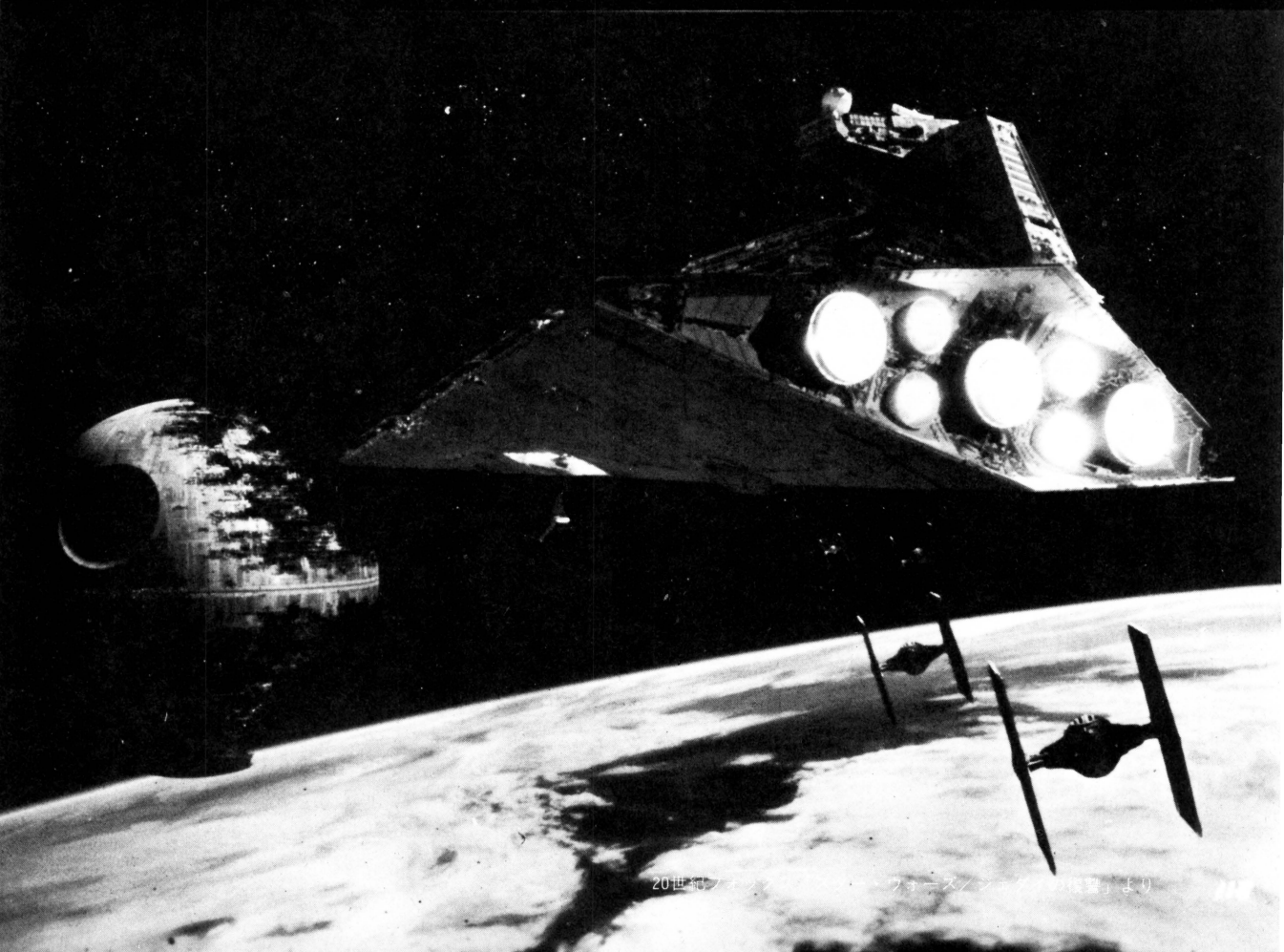
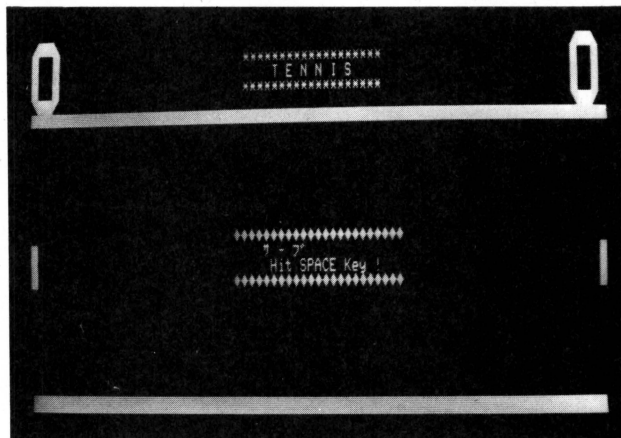
する必要があります。

以上，2系統の命令，テクニックをマスターする必要があります。

ところで，この“GAMINGへの招待”各シリーズは完全に独立しています。ですから，もしあなたの気に入ったシリーズがありましたら，お暇な折，目をむけてやってください。もしかしたら，なにか参考になるかもしれません。月刊「マイコン」誌でも，現在連載中です（宣伝，宣伝，ゴメン！）

それでは，1件落着したところで“SPACE WAR”編，クローズすることに致しましょう。バイチャ。

元祖"テニス・ゲーム"に挑戦

20世紀フォックス「スター・ウォーズ/ジェダイの復讐」より

元祖「テニス・ゲーム」に挑戦
オリエンテーション

ごあいさつ

恥ずかしながら，不肖，ワタクシ，この度

GAMING への招待

と題しまして，文を連載することになりました。チョッと気になるタイトル—— 一体何をやろうとしているのか？ ハイ。

とにかくカタイことは抜きに，GAMEで遊んで，イヤ，GAMEでも作って御一緒に楽しんでみましょう——という主旨でして，ハイ。

- **マイコンを始めたばかりのあなた！**
- **BASICの入門は一通り済んだが，そこで足踏みしているあなた！**
- **動きのあるリアル・タイム・ゲームを作りたいが，作り方がわからないで困っているあなた！**
- **すでにマシン語をものにしたあなた——**
アリャ，チョと困ったな。でも，マシン語の話題も取り上げますから，御一緒に付き合ってください。そんなあなたと，いろんな分野の，いろんなレベルのGAMEを作っていきたいと思います。そう，この小文は，GAMEを題材とした

ソフト・テクニックへのアプローチ

なのです。

S君の一人言

世の中，いつの頃かマイコン，マイコンて騒いで，
「あなたもマイコン時代に生きられるか！」
なんてテレビでも囃子たててさ。そりゃ，ドキッとするさ。あせりもしますわ。そんでもってこりゃ取り残されちゃいけないと思ってさ，大枚持ってマイコン・ショップに行きますわ。すると若僧の店員が出て来てさ，
「どのような目的でお始めになりますか？」
とくら。どんな目的？ そんなのわかるわけないでしょ。こっちとくら，コンピュータのコの字も知らない。
「それなら，この機種がよろしいでしょ，今，一番出回っている機械ですから」

そんでもって**ベーシック**とかいう本をシコシコ読んで勉強しますわね。そりゃ，**PRINT**とか**LET**とかわかりますよ。でも入門書は読んでみましたよ。でも雑誌にのってるアレ，なにさ。サッパリ，チンプンカンプン。そりゃ凄いと思いますよ。オレだって作りたいよ，あんなプログラムを。自由に使いこなしたいよ。大枚はたいた自分のマシーンだもの。

テレビ・ゲームに挑戦

あなたはこのS君の一人言を読まれて，どのような感想をお持ちになったでしょうか？

私は以前，マイコン・クラブにおいて初心者向けの

講習を何度か行ったことがありました。そして，一通りＢＡＳＩＣの文法はマスターしたものの，そこで行きづまっている人が意外と多いのに驚きました。プログラミングなんて，ちょっとしたきっかけで出来るようになるものです（マシン語を含めて）。

そこでそんなＳ君の期待に応えるべく，今回から４回にわたって**テレビ・ゲームの元祖，一世を風靡した？**　あの「テニス・ゲーム」を作ってみたいと思います。あなたもこの機会に，ぜひ**動きのあるリアル・タイム・ゲーム**の作り方をマスターしてください。

その前に―――

読者への挑戦

この節は，初心者の方は読み飛ばしてください。とくにＢＡＳＩＣの腕に覚えのある読者を歓迎します。

> **問題）**　画面が40×25モードに設定されているとします。この画面上を“♥”がランダムに動くプログラムを，**ＢＡＳＩＣ１行だけ**で作ってください。
>
> コメント）当然上下左右のはみ出しチェックが必要です。もしあなたがＢＡＳＩＣに自信を持っているなら，ＩＦ～ＴＨＥＮ～ＥＬＳＥの多重ネスティングをするような野暮なプログラムは書かないでください。

この問題を解くには，当然四つのチェックが必要です。そのためマルチ・ステートメントの使用を余儀なくされます。それは仕方がないでしょう。そして本来ならＩＦ文が四つ必要になります。ということは，いくらマルチ・ステートメントを多用しても４行以上のプログラムになってしまいますね。しかし―――。

この問題，実はこの原稿を書いているとき，ふと思いついたものです。今回の絵を動かすテーマに関係があったので，取り上げてみました。一つの**プログラミング・パズル**として考えてみてください。答は，この小文のどこかにあります。

二つのテクニック

話が横道にそれてしまいました。元に戻しましょう。

さて「テニス・ゲーム」を作るには，**ボールやラケットを動かす**必要があります。読者の中には「どうやって絵を動かすのだろう？」と不思議に思っている方があるかもしれません。実はリアル・タイム・ゲームを作る場合，**絵を動かすだけではダメ**なのです。それには二つの問題があって，

①　絵を動かす

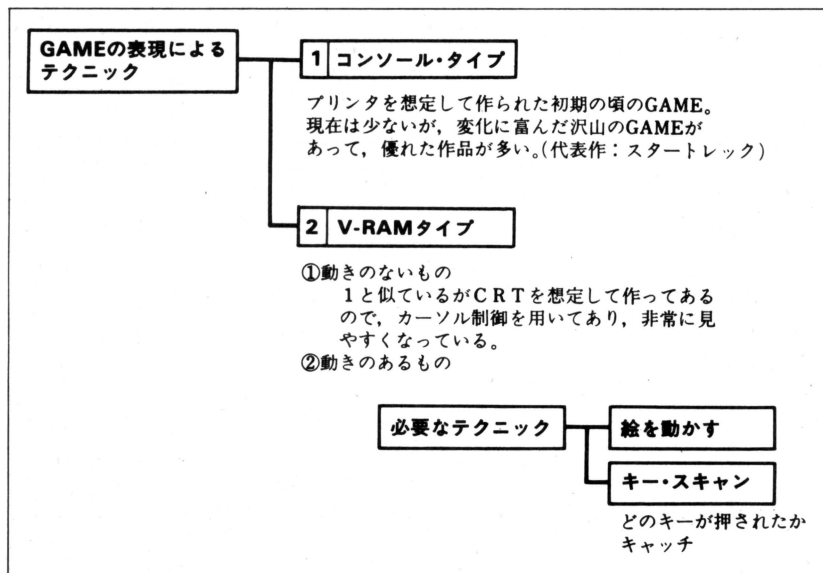
②　キー・スキャン

の二つのテクニックをものにする必要があります（**第２－１図**）。事実，昔は①のテクニックだけを使ったゲームもありました。それはそれなりに面白かったのですが，そこに②のテクニックが加わったとき，**面白さが数倍化**したものです。

《第２－１図》ＧＡＭＥのテクニック

たぶん初心者の方には何のことかピンとこないと思います。そこで順にみていくことに致しましょう。

お断わり

これから具体的に，「テニス・ゲーム」を作って行くことになります。そしてそこでは考え方を中心に説明していきたいと思います。それゆえ言語はそれを実現するための手段でしかないわけですから，ＢＡＳＩＣであろうとマシン語であろうと構わないわけです。も

ちろん機種は何であっても構いません。

しかし――――。

一応具体的にプログラム・リストは表示したいと思います。そこでその都合上，機種はＰＣ－8001に絞りたいと思います。その理由は，私がＰＣ－8001しか持っていないためで，他意はありません。

言語は一般的なＢＡＳＩＣで説明していきたいと思います。その際，他機種の方が困らないようＰＣ独自の方言については，その都度言及したいと思います。

さて，それでは――。

GAMEエリアを作る

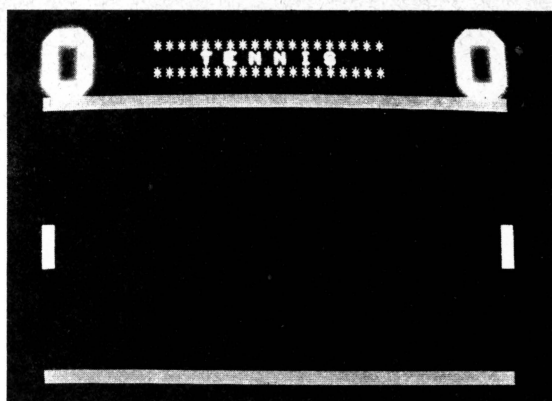
プログラムというのは，難しく考える必要はありません。あなたは何が出来ますか？

「ＰＲＩＮＴ文くらい知っている」

結構！　ＦＯＲ～ＮＥＸＴも知っている――なお結構。なにも絵を動かすことに執着する必要はないんじゃないですか。

ＰＲＩＮＴ文を知っているなら，まず「テニス・ゲーム」の画面を作ってみましょう。知っている知識をフルに活用してみましょうよ！　画面さえ作ってしまえば，**次のステップへの意欲**が湧いてくるじゃありませんか。

写真１を見てください。これが私の作った画面です。ＧＡＭＥ開始前なので，得点は０対０にセットされています。それにしてもセコイ画面ですね。しかしＰＣ－8001を使用したため色が出せますので，**配色**には気を配りました。ぜひカラー・モニタで御覧になってください。

さて，これから皆さん自身がＧＡＭＥを作っていくわけです。ですからどのような画面配置にしようと，どのようにプログラムを組もうと自由なわけです。ぜひ**御自分で，御自分のテレビ画面**を作ってみてください。最低でもカーソル制御とＰＲＩＮＴ文を知っていれば，ゲーム・エリアは作れるでしょう。

私の場合，あとあとのことを考えてプログラムをいくつかのサブ・ルーチンで構成してみました。また得点表示は頻繁におこなわれますので，２次元配列を用いました。それを次に説明してみたいと思います（**リスト２－１**）。

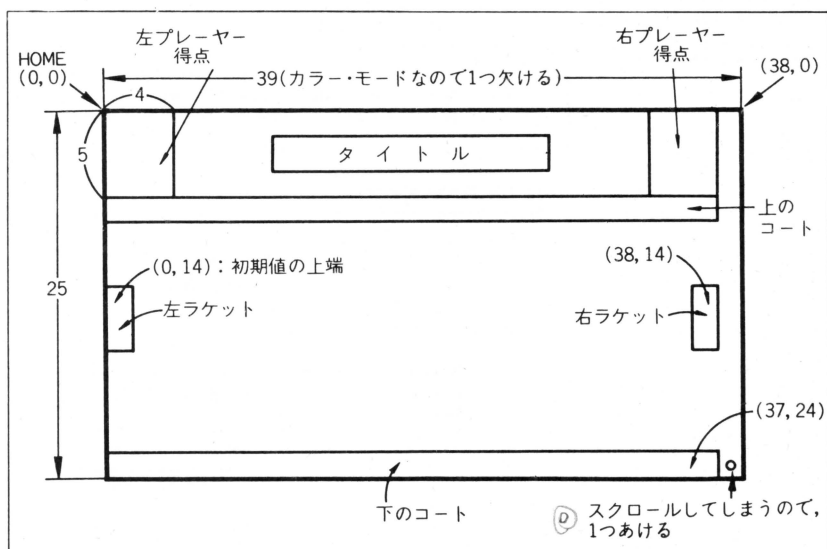
《写真１》テニスゲーム画面

プログラムの説明

画面は40×25に設定されています(**第２－２図**)。他機種の方は**第２－３図**により，ＰＣ－8001のＬＯＣＡＴＥ（カーソル制御）の使い方を見ておいてください。

① 1300～1390が画面を表示する部分です。

まず1310で画面をクリアします。次に1100のサブ・ルーチンを２回Ｃａｌｌして，左右のプレーヤーの得点を表示します。1330と1335は水平ラインを引いているところで，ＳＥＮ＄には9400の水平線が読み込まれています（1040で）。

1340～1344でタイトルを表示し，1400，1500のサ

《第２－２図》画面の設定

《リスト2－1》GAME用エリアのエリアの表示

```
10 REM ************************
20 REM *  ｶﾝﾀﾝ TENNIS GAME    *
30 REM *       by K. ﾂｶｺﾞｼ    *
40 REM *        in FORESIGHT  *
50 REM *    <81.8.31-9.??>    *
60 REM ************************
99 '
100 REM ﾒｲﾝ ﾙｰﾁﾝ
110 WIDTH 40,25:CONSOLE 0,25,0,1:COLOR 7,0,0:PRINT CHR$(12)
120 GOSUB 1000 'ｼｮｷｾｯﾃｲ
130 GOSUB 1200 'ｼｮｷｾｯﾃｲ for RE-PLAY
140 GOSUB 1300 'ｶﾞﾒﾝ
998 GOTO 998
999 '
1000 REM ｼｮｷｾｯﾃｲ
1010 DIM TEN$(7,4)
1030 FOR I=0 TO 7
1031   FOR J=0 TO 4
1032     READ TEN$(I,J)
1033   NEXT J
1034 NEXT I
1040 READ SEN$
1090 RETURN
1099 '
1100 REM ﾃﾝ ﾉ ﾋｮｳｼﾞ
1110 X=0:IF LR$="r" THEN X=34
1120 FOR I=0 TO 4
1130   LOCATE X,Y+I:PRINT TEN$(TEN,I)
1140 NEXT I
1190 RETURN
1199 '
1200 REM ｼｮｷｾｯﾃｲ for RE-PLAY
1210 LTEN=0:RTEN=0:LRA=14:RRA=14
1290 RETURN
1299 '
1300 REM ｶﾞﾒﾝ
1310 PRINT CHR$(12)
1320 COLOR 6:LR$="l":TEN=LTEN:GOSUB 1100
1325         LR$="r":TEN=RTEN:GOSUB 1100
1330 COLOR 1:LOCATE 0,5:PRINT SEN$;
1335         LOCATE 0,24:PRINT SEN$;
1340 COLOR 2:LOCATE 9,1:PRINT "********************";
1342 COLOR 7:LOCATE 9,2:PRINT "     T E N N I S    ";
1344 COLOR 2:LOCATE 9,3:PRINT "********************";
1350 GOSUB 1400 'ﾋﾀﾞﾘ ﾗｹｯﾄ
1360 GOSUB 1500  'ﾐｷﾞ ﾗｹｯﾄ
1390 RETURN
1399 ;
1400 REM ﾋﾀﾞﾘ ﾗｹｯﾄ
1410 COLOR 3
1420 LOCATE 0,LRA  :PRINT "■";
1430 LOCATE 0,LRA+1:PRINT "■";
1440 LOCATE 0,LRA+2:PRINT "■";
1490 RETURN
1499 ;
1500 REM ﾐｷﾞ ﾗｹｯﾄ
1510 COLOR 3
1520 LOCATE 37,RRA  :PRINT "■";
1530 LOCATE 37,RRA+1:PRINT "■";
1540 LOCATE 37,RRA+2:PRINT "■";
1590 RETURN
1599 ;
9000 DATA "▄▄▄"
9010 DATA "█ █"
9020 DATA "█ █"
9030 DATA "█ █"
9040 DATA "▀▀▀"
9050 DATA " ▄█ "
9060 DATA "  █ "
9070 DATA "  █ "
9080 DATA "  █ "
9090 DATA " ▄█▄"
```

注記：

- 110〜140：画面モードを初期化　各サブルーチンをcallする
- 998：無限ループ，ENDの代わり。ENDだと，「OK」が出てスクロールしてしまう。
- 1010：配列の定義
- 1030〜1040：READ文を使って，TEN$(○，○)，SEN$にデータを読み込む
- 1110：LR$＝"l"ならX＝0，LR$＝"r"ならX＝34に
- 1120〜1140：得点の表示
- 1210：変数の初期化，変数表参照
- 1320：左のプレーヤーの得点表示
- 1325：右のプレーヤーの得点表示
- 1330：上の水平線
- 1335：下の水平線
- 1420〜1440：左のプレーヤーのラケットを表示する　LRAにラケットの上端のY座標が入っている

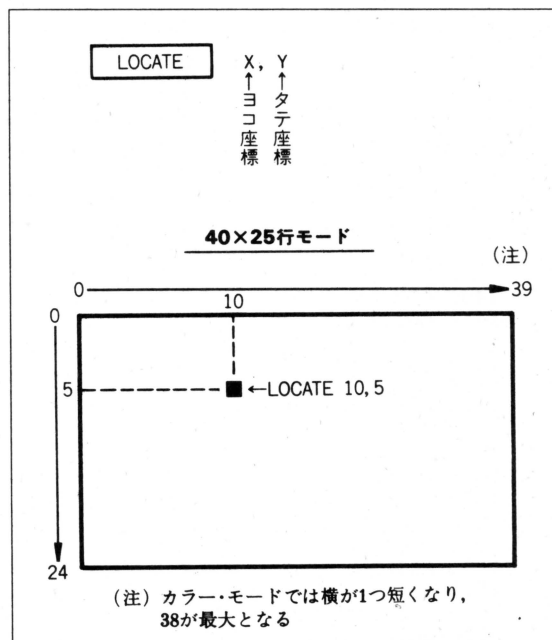
```
9100 DATA "◢■■◣"
9110 DATA "■  ■"
9120 DATA "   ◢"
9130 DATA "  ◢ "
9140 DATA "◢■■■"
9150 DATA "◢■■◣"
9160 DATA "■  ◤"
9170 DATA "  ■ "
9180 DATA "■  ◣"
9190 DATA "◥■■◤"
9200 DATA "◢   "
9210 DATA "■ ■ "
9220 DATA "■ ■ "
9230 DATA "◥■■◣"
9240 DATA "  ■ "
9250 DATA "■■■◣"
9260 DATA "■   "
9270 DATA "◥■■◣"
9280 DATA "   ■"
9290 DATA "◥■■◤"
9300 DATA "◢■■◣"
9310 DATA "■   "
9320 DATA "■■■◣"
9330 DATA "■  ■"
9340 DATA "◥■■◤"
9350 DATA "◢■■■"
9360 DATA "◤  ■"
9370 DATA "   ■"
9380 DATA "   ◢"
9390 DATA "  ◢ "
9400 DATA "■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■"
```

《第２－１表》変数表

TEN$(○, ○) ↑得点 0～7 上からの位置 0～4	得点表示用データを入れておく２次元配列
SEN$	コートの水平線用データ
LR$	1100のサブルーチンの入力引数で，左のプレーヤーなら“l”，右のプレーヤーなら“r”を入れてCallする。
LTEN	左のプレーヤーの得点
RTEN	右のプレーヤーの得点
LRA	左のプレーヤーのラケットの上端の座標
RRA	右のプレーヤーのラケットの上端の座標

ブ・ルーチンをCallして左右のラケットを表示します。

② 次に得点表示の1100～1190のサブ・ルーチンを見てみましょう。

このルーチンは変数TENに表示したい点数（0～7），文字変数LR$に“L”(左のプレーヤー)か“R”(右のプレーヤー）を入れてCallすると，それぞれの位置に得点が表示されます。

《第２－３図》PC－8001による画面制御

この得点を表示するのにTEN$（○，○）という２次元配列を用いました。第２－４図で使い方を理解してください。

★

その他はプログラムに詳しいコメントを付けましたから，わかると思います。もし知らない命令があったら，マニュアルと見比べて完全に理解しておいてください。

《第２－４図》得点の表示

絵を動かす

さてこれでGAMEエリアの画面が出来上がりました。第２章ではいよいよ**ボール**や**ラケット**を登場させて，**絵を動かすしくみ**を解剖していくことに致しましょう。

絵を動かす原理は非常に簡単です。たとえば**リスト２**はたった１行のプログラムですが，これだけで♥が画面上をランダムに動きまわります。またこれを**リスト３**のように変更すれば，足跡を残しながら動きまわり，奇麗な模様が出来上がります（**写真２**）。

その原理は――？

第２章をお楽しみに。

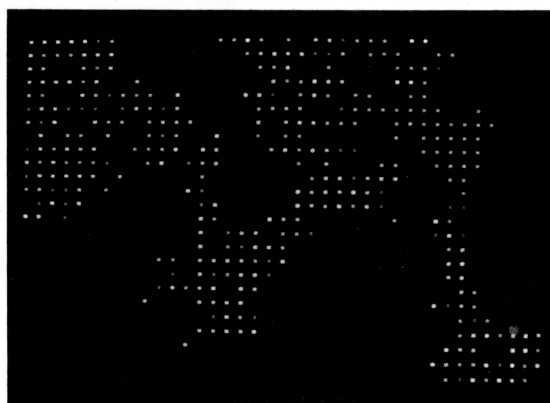
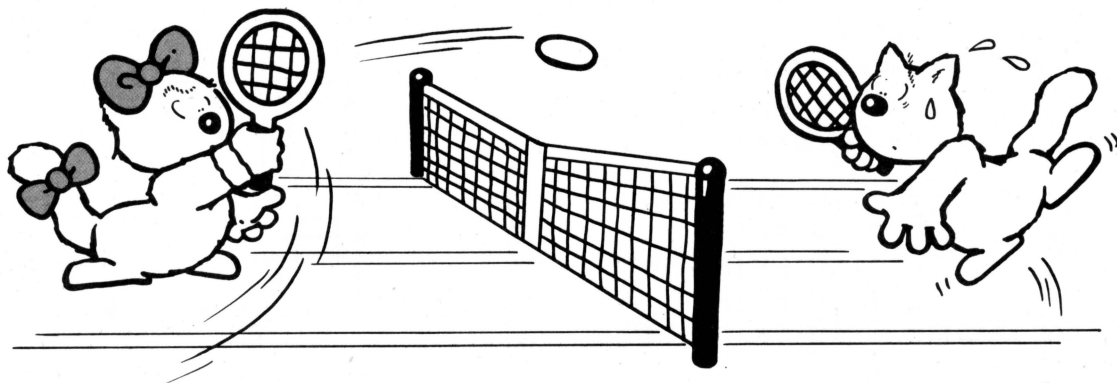
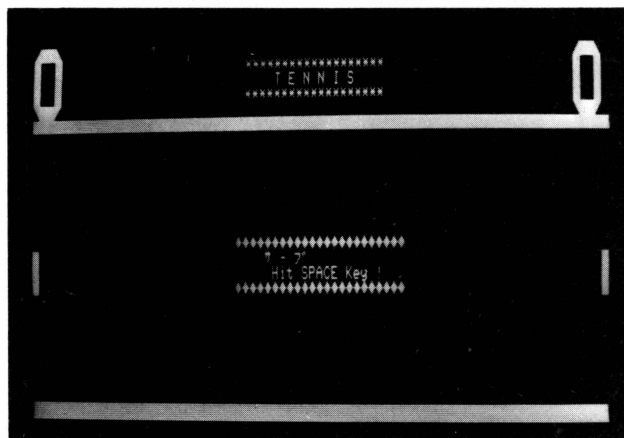
《写真２》♥がランダムに模様を作る

《リスト２－２》画面上をランダムに♥を動かす（ＩＦ文が全く使われていないのは驚異！）

```
10 LOCATE X,Y:PRINT " ";:X=X+INT(RND(1)*3-1):Y=Y+INT(RND(1)*3-1):X=X+(X>-1)-(X<3
9):Y=Y+(Y>-1)-(Y<25):LOCATE X,Y:PRINT "♥";:GOTO 10
```

《リスト２－３》♥がランダムな模様を作る（カラーモードに設定してから見てください）

```
10 COLOR RND(1)*8:LOCATE X,Y:PRINT "・";:X=X+INT(RND(1)*3-1):Y=Y+INT(RND(1)*3-1):
X=X+(X>-1)-(X<39):Y=Y+(Y>-1)-(Y<25):LOCATE X,Y:PRINT "♥";:GOTO 10
```

第2章 元祖「テニス・ゲーム」に挑戦

図形を動かす

はじめに

かつて私の知人で，買ったばかりのマイコンを**数か月もたたないうちに買いかえた**人がいました。しかも売ったマシンはソフトも大量に出回っているベスト・セラー機で，買いかえたマシンがその逆のあまり出回っていないものだったのです。

その人曰く，

「ベスト・セラー・マシンはソフトが豊富で，自分で作らなくても待っていれば必ず誰かが発表してくれる。これじゃプログラムを作る気にもなれない」

わかるような気もします。

とかくベスト・セラー機をお持ちの方は，発表済みのソフトに甘え，自作する気力を失なっているようです。しかし少なくともこの小文を読もうとされている方は，**創作の意欲に燃えている人**だと思います。

ですから第1章で紹介しましたGAMEエリアにしても，ただ私の例をそのまま打ち込むのでなく，ぜひ御自分の方法でチャレンジしていただきたいと思います。その上で第2章の記事をお読みください。

第2章では絵の動かし方の原理を調べていきたいと思います。それをマスターしたとき，あなたは「マイコンGAME」の，否,「**マイコン・プログラミングのエキスパート**」への第一歩を踏み出したことになるでしょう。

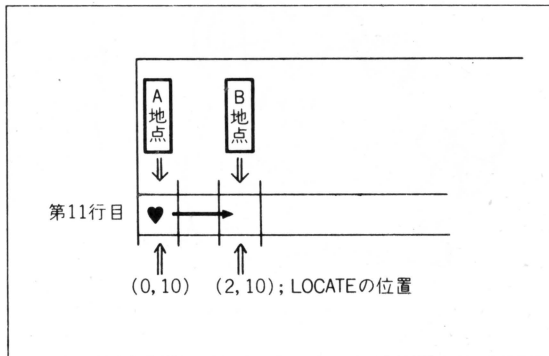
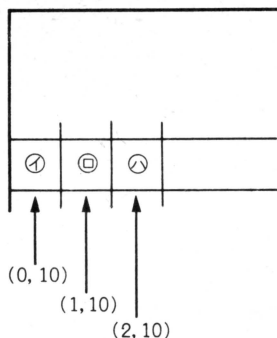
《リスト2－4》♥を動かす

```
10 LOCATE 0,10:PRINT "♥"
20 LOCATE 0,10:PRINT " "
30 LOCATE 1,10:PRINT "♥"
40 LOCATE 1,10:PRINT " "
50 LOCATE 2,10:PRINT "♥"
60 LOCATE 2,10:PRINT " "
70 GOTO 10
```

図形を動かす原理

まず**リスト2－4**を御覧ください。

これは**第2－5図**の**A地点**から**B地点**まで♥を動かそうとして作ったプログラムです。原理はアホらしい程簡単で，ネオン・サインのように♥をつけたり消したりしながら右に動かしていきます（**第2－6図**）。

それでは実際に**リスト2－4**をキー・インし，プログラムを走らせてみてください。

《第2－5図》A地点からB地点まで♥を動かす

①	㋑に♥を書く	LOCATE 0,10 : PRINT "♥"
②	㋑の♥を消す	LOCATE 0,10 : PRINT "　"
③	㋺に♥を書く	LOCATE 1,10 : PRINT "♥"
④	㋺の♥を消す	LOCATE 1,10 : PRINT "　"
⑤	㋩に♥を書く	LOCATE 2,10 : PRINT "♥"
⑥	㋩の♥を消す	LOCATE 2,10 : PRINT "　"

①〜⑥を繰り返せば，ネオン・サインのように
♥が㋑・㋺・㋩の順に移っていくように見えるはず。

《第２－６図》♥を動かすしくみ

いかがでしたか？

「速すぎて，動きが良くわからない！」

とお感じになった人が多いのではないでしょうか？ごもっともです。それでは，**リスト２－４**の方法は絵の動かし方として不適当だったでしょうか？

否！

リスト２－４の方法は，現在マイコンを使ってCRT上に図形を動かそうとするとき，**最もポピュラーな手法**なのです。まとめておきましょう。

図形を動かすテクニック１

① 図形を表示する。
② 図形を消去する。
③ 次の位置に図形を表示する。
④ また図形を消去する。
⑤ 以上を繰り返す。

以上はまさに**ネオン・サインの原理**そのものです。

目にも止まるタイマーの威力

さて原理的には正しくても，**リスト１**のままでは視覚的には不都合です。そこでタイマーとかディレイと

《リスト２－５》タイマーの威力

```
10 LOCATE 0,10:PRINT "♥":GOSUB 80
20 LOCATE 0,10:PRINT " ":GOSUB 80
30 LOCATE 1,10:PRINT "♥":GOSUB 80
40 LOCATE 1,10:PRINT " ":GOSUB 80
50 LOCATE 2,10:PRINT "♥":GOSUB 80
60 LOCATE 2,10:PRINT " ":GOSUB 80
70 GOTO 10
80 FOR T=0 TO 50:NEXT:RETURN
```

《第２－７図》タイマーの原理

呼ばれるテクニックが登場します。

リスト２－５を御覧ください。

リスト２－４との違いは，80行のサブルーチンが加わったことです。それでは80行のサブ・ルーチンでは何が行なわれているのでしょうか？

答――何も行なわれていません（**第２－７図**）。

ただ，FOR〜NEXTにより数をかぞえているだけです。その代わり**少し時間がかかります**――ここが重要で，時間がかかるため♥の移動も**リスト２－４**に比べ時間がかかり，結局人間の**目にもとまる速さ**に落ち着くわけです。

それでは**リスト２**をキー・インし，自分の目で確かめてください。タイマーの威力がわかるでしょう。

図形を動かすテクニック２

図形の動きが速すぎるときは，**タイマー**を用いる。

チャレンジ・コーナー

一応，絵の動かし方，タイマーの使い方がわかったところで，これらを使って簡単なプログラムに挑戦し

てみましょう。

> **チャレンジ1**
> ♥を画面の第10行の左端から右端まで動かすプログラムを作ってください。

原理は今までと同じです。しかし**リスト2－4**や**リスト2－5**に比べ，今度は39個も♥を書いたり消したりしなければなりません(カラー・モードの場合)。そこで横座標を表示するのに，**変数**を用いるのがポイントとなるでしょう。

解答例としては，**リスト2－6**を参考にしてください。

《リスト2－6》横座標に変数を用いる

```
300 FOR X=0 TO 38
310    LOCATE X,10:PRINT "♥":GOSUB 1000
320    LOCATE X,10:PRINT " ":GOSUB 1000
330 NEXT X
340 GOTO 300
1000 FOR T=0 TO 10:NEXT:RETURN
```

> **チャレンジ2**
> ♥を画面の端にそって，グルグルまわるプログラムを作ってください。

チャレンジ1がわかれば，座標に変数を用いれば良いことがわかるでしょう。いろいろなやり方があると思いますが，入門コーナーということで**リスト2－7**のような解答を考えてみました。ちょっとくふうすれば，もっと短く出来ます。

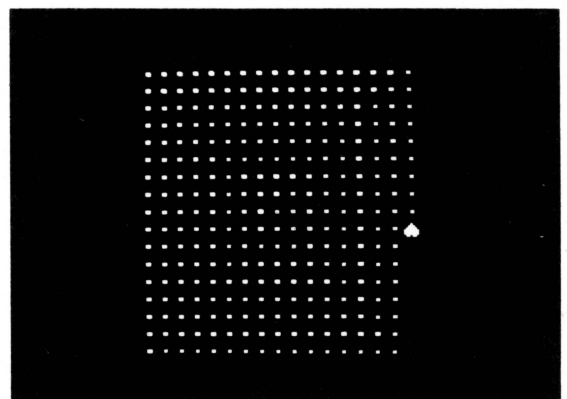
《リスト2－7》♥をグルグルまわす

```
10 FOR X=0 TO 38
20    LOCATE X,0:PRINT "♥";:GOSUB 180
30    LOCATE X,0:PRINT " ";:GOSUB 180
40 NEXT X
50 FOR Y=0 TO 23
60    LOCATE 38,Y:PRINT "♥";:GOSUB 180
70    LOCATE 38,Y:PRINT " ";:GOSUB 180
80 NEXT Y
90 FOR X=38 TO 0 STEP -1
100    LOCATE X,23:PRINT "♥";:GOSUB 180
110    LOCATE X,23:PRINT " ";:GOSUB 180
120 NEXT X
130 FOR Y=23 TO 0 STEP -1
140    LOCATE 0,Y:PRINT "♥";:GOSUB 180
150    LOCATE 0,Y:PRINT " ";:GOSUB 180
160 NEXT Y
170 GOTO 10
180 FOR T=0 TO 10:NEXT:RETURN
```

> **チャレンジ3**
> ♥をうず巻状に動かすプログラムを作ってください。

前の二つに比べ，ちょっと難しいかもしれません。そこでどうせ示すならと思って，ちょっと凝った解答を作ってみました。**リスト2－8**がそれです。これは乱数でカラーの色を変えながら，しかも足跡を残しながらうず巻を表示していきます。画面いっぱい表示が終わると，今度はうずを巻きながら，♥が足跡を消していきます（**写真3**）。簡単なプログラムですが，配色が奇麗なので，ぜひカラー・モニタで見てください。

またプログラムそのものは，単純な構成になっていますから，**リスト2－8**を読んでみてください。

以上で今回のメイン・テーマである「図形の動かし方」がおわかりいただけたと思いますので，次にいよいよ第1章で作ったテニス・コート場内で，ボールを動かすことに致しましょう。

《写真3》カラーうず巻

ボールを動かす問題点

さて，ボールを動かそうとするとき，今までの例と違っていくつかの問題点があるのに気が付きます。

それらを列挙してみると，

《リスト2－8》カラーうず巻

```
10 WIDTH40,25:CONSOLE0,25,0,1:PRINT CHR$(12)
20 A$="・":GOSUB 50
30 A$=" ":GOSUB 50
40 GOTO 20
50 X=18:Y=12
60 FOR I=1 TO 23 STEP 2
70   COLOR RND(1)*7+1
80   FOR J=1 TO I
90     LOCATE X,Y:PRINT "♥";:GOSUB 330
100      LOCATE X,Y:PRINT  A$;:GOSUB 330
110      Y=Y-1
120    NEXT J
130    COLOR RND(1)*7+1
140    FOR J=1 TO I
150      LOCATE X,Y:PRINT "♥";:GOSUB 330
160      LOCATE X,Y:PRINT  A$;:GOSUB 330
170      X=X+1
180    NEXT J
190    COLOR RND(1)*7+1
200    FOR J=1 TO I+1
210      LOCATE X,Y:PRINT "♥";:GOSUB 330
220      LOCATE X,Y:PRINT  A$;:GOSUB 330
230      Y=Y+1
240    NEXT J
250    COLOR RND(1)*7+1
260    FOR J=1 TO I+1
270      LOCATE X,Y:PRINT "♥";:GOSUB 330
280      LOCATE X,Y:PRINT  A$;:GOSUB 330
290      X=X-1
300    NEXT J
310 NEXT
320 RETURN
330 FOR T=0 TO 5:NEXT:RETURN
```

① ボールをななめに動かすにはどうするか？

② しかもその方向はバラバラ。

③ 仮にななめに動かすことに成功しても，そのまま放っておくと，いずれコートをつき破り，やがて画面の端に到達し，Illegal function call を起こすことになる（第2－8図）。

④ またラケットに当った時や，受けそこなった時はどうする？

等々いろいろあります。順に解決していきましょう。

四つの異なる処理

まずボールを表示する際のカーソル制御ですが，当然変数を用います。XBLL，YBLLを使うことにします（第2－2表）。

その上で前節の問題点ですが，まず②を整理してみましょう。第2章は入門ということもあって第2－9図のように4方向に分類してみました。そして4方向を数字の1～4で記憶することにし，変数CRSにストアしておくことにします。なお蛇足ながら，1～4の決め方は「解析幾何」の象限に対応しています。本当は0～3を用いる方が，あとで配列を使う時にメモリの節約になるのですが，あくまで私の好みで決めさせていただきました。

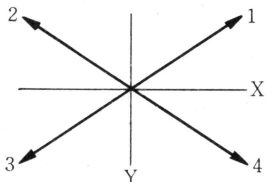
そこで①についてですが，右上にボールを進ませるには，原理的に第2－10図のようにすれば良いことがわかります。さらに同じ理由から，他の3方向についてもまとめると，第2－11図のようになります。

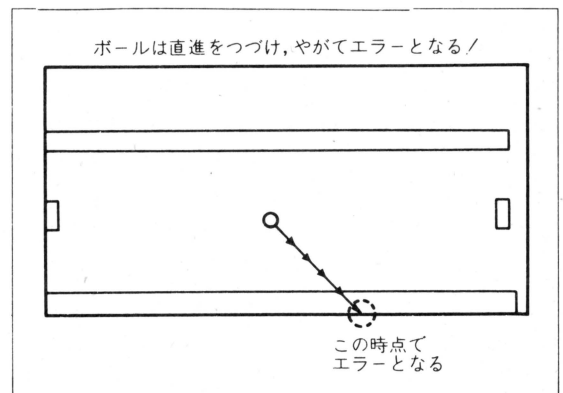
以上の考察から，

ボールを4方向に動かすには，

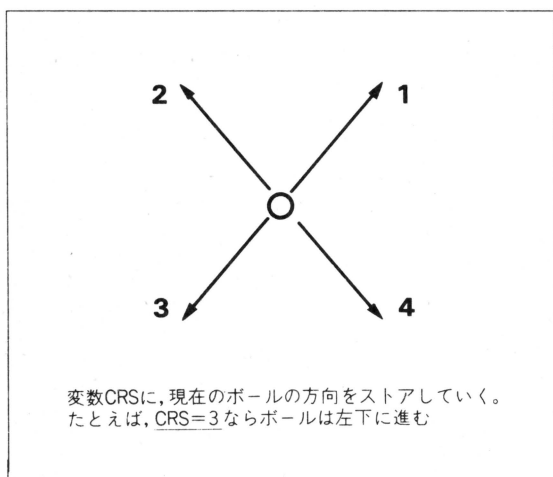
四つの異なる処理が必要になる

ように見えます。実際，私が昔見た「ブロックくずし」のプログラムでは，四つの方向を処理するのに，四つのサブ・ルーチンを用いていました。これは，

ムダだ！

面倒臭い！

と思いませんか？

《第2－8図》ボールがとまらず……

《第2－2表》変数表（第1章のつづき）

XBLL YBLL	ボールの現在値
XMOV(4) YMOV(4)	ボールの変位
CRS	ボールの方向を表わす(1～4) 2 1 X 3 Y 4
XNXT YNXT	ボールのつぎの位置

《第2－9図》ボールの4方向に数字を対応させる

思ったら解決策を見つけることです。

解決！

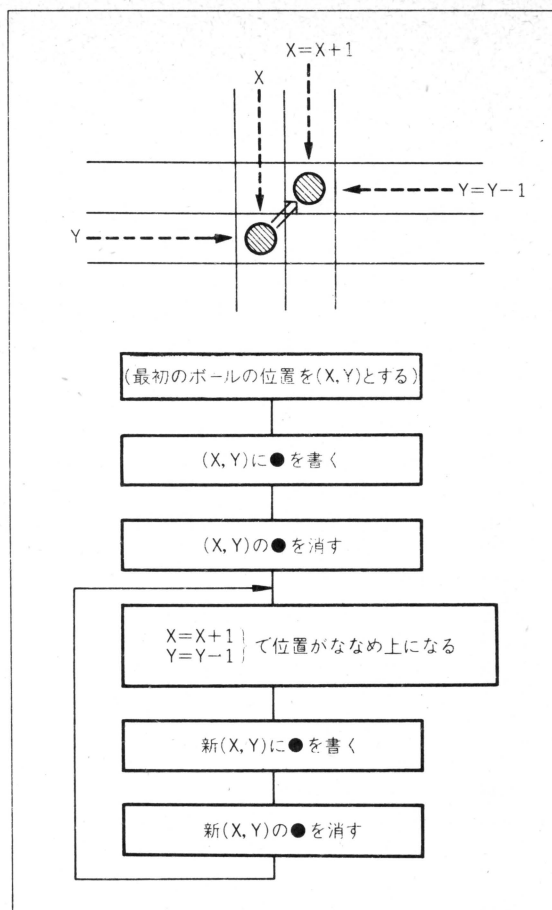
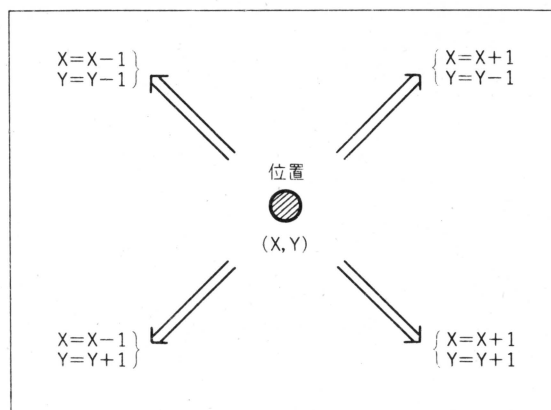
いろいろな解決法があると思いますが，私は

XMOV(4)，YMOV(4)

という二つの配列を使ってみました。まず9410行のデータを用意します(**リスト2－9**)。第1章1010にリストのように追加して配列を定義し，1050～1054のようにデータを読み込みます。すると配列の値は**第2－12図**のようになりますから，**第2－13図**の計算を行えば，

たった一つのやり方で，

四つの異なる条件を処理する

ことが出来ることがわかると思います。これで②が解決いたしました。

《第2－10図》ボールを右上に進めるには

《第2－11図》ボールを4方向に動かす

反射の処理

さて残りは③と④です。

このうち④については，**ラケットの動きとの絡み**があるので，本章に限っては，③のコート同様にラケットに当たろうとはずれようと，画面の左右端に来たときは，**すなおにボールを反射させる**ことにします。

《リスト2－9》第2章までの「テニス」(1600行以降のサブルーチンが第1章に追加。その他，少し変更・追加がある)

```
10 REM ************************
20 REM *  ｶﾞﾝｼﾞ TENNIS GAME    *
30 REM *       by K. ﾂｶｺﾞｼ     *
40 REM *         in FORESIGHT  *
50 REM *    <81.8.31-9.??>     *
60 REM ************************
99 '
100 REM ﾒｲﾝ ﾙｰﾁﾝ
110 WIDTH 40,25:CONSOLE 0,25,0,1:COLOR 7,0,0:PRINT CHR$(12)
120 GOSUB 1000 'ｼｮｷｾｯﾃｲ
130 GOSUB 1200 'ｼｮｷｾｯﾃｲ for RE-PLAY
140 GOSUB 1300 'ｶﾞﾒﾝ
150 XBLL=18:YBLL=14:CRS=1:COLOR 7:LOCATE XBLL,YBLL:PRINT "●";
160 GOSUB 1600:GOTO 160' ﾎﾞｰﾙ ｦ ｳｺﾞｶｽ
998 GOTO 998
999 '
1000 REM ｼｮｷｾｯﾃｲ
1010 DIM TEN$(7,4),XMOV(4),YMOV(4)
1030 FOR I=0 TO 7
1031   FOR J=0 TO 4
1032     READ TEN$(I,J)
1033   NEXT J
1034 NEXT I
1040 READ SEN$
1050 FOR I=1 TO 4
1052   READ XMOV(I),YMOV(I)
1054 NEXT I
1090 RETURN
1099 '
1100 REM ﾃﾝ ﾉ ﾋｮｳｼﾞ
1110 X=0:IF LR$="r" THEN X=34
1120 FOR I=0 TO 4
1130   LOCATE X,Y+I:PRINT TEN$(TEN,I)
1140 NEXT I
1190 RETURN
1199 '
1200 REM ｼｮｷｾｯﾃｲ for RE-PLAY
1210 LTEN=0:RTEN=0:LRA=14:RRA=14
1290 RETURN
1299 '
1300 REM ｶﾞﾒﾝ
1310 PRINT CHR$(12)
1320 COLOR 6:LR$="l":TEN=LTEN:GOSUB 1100 'ﾋﾀﾞﾘ PLAYER ﾄｸﾃﾝ
1325         LR$="r":TEN=RTEN:GOSUB 1100 'ﾐｷﾞ  PLAYER ﾄｸﾃﾝ
1330 COLOR 1:LOCATE 0,5:PRINT SEN$;
1335        LOCATE 0,24:PRINT SEN$;
1340 COLOR 2:LOCATE 9,1:PRINT "********************";
1342 COLOR 7:LOCATE 9,2:PRINT "    T E N N I S     ";
1344 COLOR 2:LOCATE 9,3:PRINT "********************";
1350 GOSUB 1400 'ﾋﾀﾞﾘ ﾗｹｯﾄ
1360 GOSUB 1500  'ﾐｷﾞ ﾗｹｯﾄ
1390 RETURN
1399 ;
1400 REM ﾋﾀﾞﾘ ﾗｹｯﾄ
1410 COLOR 3
1420 LOCATE 0,LRA  :PRINT "■";
1430 LOCATE 0,LRA+1:PRINT "■";
1440 LOCATE 0,LRA+2:PRINT "■";
1490 RETURN
1499 ;
1500 REM ﾐｷﾞ ﾗｹｯﾄ
1510 COLOR 3
1520 LOCATE 37,RRA  :PRINT "■";
1530 LOCATE 37,RRA+1:PRINT "■";
1540 LOCATE 37,RRA+2:PRINT "■";
1590 RETURN
1599 ;
```

XMOV(1)=1 YMOV(1)=−1	右上への変位（CRS=1）
XMOV(2)=−1 YMOV(1)=−1	左上　〃　（CRS=2）
XMOV(3)=−1 YMOV(1)=1	左下　〃　（CRS=3）
XMOV(4)=1 YMOV(1)=1	右下　〃　（CRS=4）

《第2−12図》ボールの変位用配列の値

《第2−13図》ボールを動かすときのカーソルの計算

《第2−14図》ボールがコートをはみ出す条件

《写真4》ボールの軌跡が残るテニス

すると残りの問題は③だけです。

以上を頭に入れた上で1600〜1690のサブ・ルーチンを見てください。1610で（XNXT，YNXT）に次のボールの位置を計算しています。その結果，ボールが反射するコートのはじに来たかを1620〜1650でチェックしています（**第2−14図**）。上下方向に反射したときは1700，左右方向については1800のサブ・ルーチンをCall して，ボールの方向を変化させています。ボールが反射しないときは今まで何回もやったように，**ボールを消して次の位置に表示**というパターンを1660〜1680で処理しています。

なお本章では**ボールの軌跡**がわかるようにしてみました。1660の“・”を“　”に変えれば，ボールの軌跡は残りません。**リスト2−9**のように入力して，RUNさせると**写真4**のようになります。黙って画面を見ていると，だんだんとコート上をボールの軌跡がうめていきます。結構面白いので，ぜひ御覧になってください。

なお1800のルーチンについては，やがてプログラムの進行に伴なって変更されることがわかるでしょう。

以上でボールは動きました。あとはラケットが動けば良いのです。そしてその動かし方も，あなたは知っているわけです。

しかし——。

ここでついに第1章で述べたリアル・タイム・ゲームの**第2の問題点**にぶつかることになります。絵の動かし方を知らなかったあなたは，**絵の動かし方**さえわかれば，「インベーダー」であろうと何であろうとリアル・タイム・ゲームの作り方がわかると思っていたのではないでしょうか？　そして今，あなたは絵の動かし方がわかりました。

しかし，

スティックの動かし方には困ったのではないでしょうか？　ただ動かすだけなら，今までの知識だけで可能です。でも，でも，スティックについては，それだけでは**ダメ**なのです。つまり，

特定のキーが押されたときだけ
絵を動かす

テクニックが必要になってくるのです。これには，

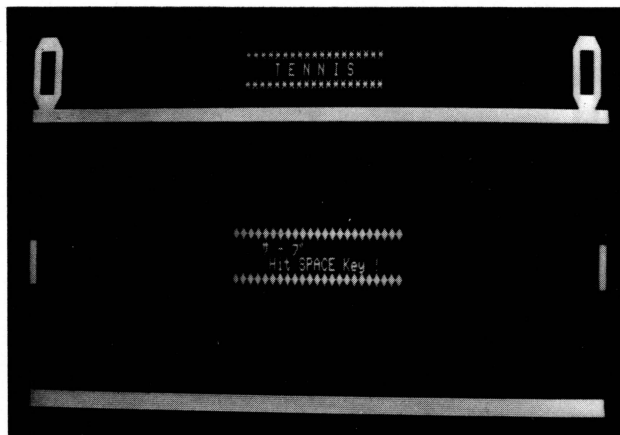
キー・スキャン

の知識が必要になってきます。

となると第3章のテーマは，

キー・スキャンに挑戦！

ということになってきます。先は短い！　頑張りましょう。

第3章 元祖「テニス・ゲーム」に挑戦
キー・スキャン

はじめに

「マイコン」って，「マイクロ・コンピュータ」のことですが，「マイ・コンピュータ」の意味も含まれていますよね（「マイコン時代」中央公論・経営問題／増刊号P.45)。当然自分で買った自分のマシンです。

自分による
自分のための
自分のマイコン

です。素直に楽しんだら良いのではないでしょうか？しかし，意外とそうでない人が多いようです。なんとか仕事や実用に使いこなそうと苦しんでいるようです。でも，たまには**遊びの要素**を取り入れてみたらいかがでしょうか？

マイコンのソフト面での活用をみてみると大きく

① 実用ソフト
② ゲーム
③ システム

の三つに分類されるようです。そして突き詰めて研究していくとどの分野も奥が深いのですが，最初の**取りつきにくさ**からみれば，③，②，①の順になるでしょう。事実，ゲームを1本作ってごらんなさい。実用ソフトなんてチョロくみえてきます。おおいにゲーム作りを楽しみましょう。

問題点―キー・スキャン

第2章ではボールを動かすのに成功しました（成功しましたか？ OKですね)。そして次にラケットを動かそうとした時，一つの壁につき当ってしまいました。

キー・スキャン

第3章では，この問題を一つ一つ解決していくことに致しましょう。

目的は，**ラケットを動かすこと**です。そしてただ動かすだけのテクニックでしたらすでにものにしました。問題は，

特定のキーが押されたときだけ
ラケットを動かしたい

のです。すると問題は，いかにしてキー・ボードの入力をキャッチするかに絞られてきます。普通この問題は，

キー・スキャン

と呼ばれています。

キー入力の二つのタイプ

N－BASICでは，キー・入力のためのいくつかのコマンドが用意されています（**第2－3表**）。これらのキー・ワードについて，もし知らないものがあれば，わからなくても結構ですから一応マニュアルに目を通してみてください。OK？

キー入力については大きく，

《第2－3表》キー入力用コマンド

キー・ワード	WAIT
INPUT	あ　り
INPUT$	〃
LINE　INPUT	〃
INKEY$	な　し

WAITつき，キー入力———①
WAITなし，キー入力———②

の二つに分けることができます。比較的ポピュラーな入力キー・ワードである，

INPUT
INPUT$
LINE　INPUT

は①のタイプに属します。実用ソフトしか作ったことのない人は，①のタイプのキー・ワードしか使ったことがないかもしれません。

①と②の違いは，①は入力があるまで次の命令を実行しないのに対し，②は入力があろうとなかろうと次の命令を実行してしまうことにあります。高級言語からみれば，①の方がポピュラーですが，意外，マシン語レベルからみれば①のルーチンは②のルーチンをベースに組み立てられています（第2－15図）。

どれを選ぶか—

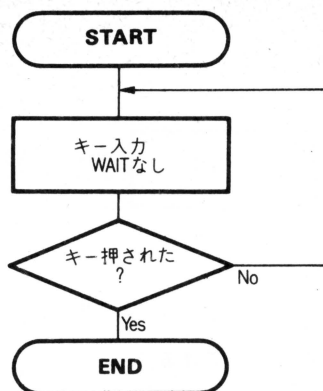
さてキー入力には二つのタイプがあることがわかったのですが，ラケットを動かすのに使えそうなのは，当然②のタイプであることがわかります。もし，①のタイプのキー入力を採用すると，キー入力までWAITがかかり，結果としてキーを一つ動かすまでボールが動かないことになります。

リアル・タイム・ゲームでは
WAITつきキー入力では不適当！

となると，第3章の問題解決のためには**第2－3表**の残りのキー・ワードが候補にのぼってきます。

結論から申し上げましょう。実は，

第2－3表のキー・ワードの中には
キー・スキャンに使えるものはない！

のです。なぜなら，キー・スキャンに使える命令は当然WAIT無しのものでなければ困るが，**WAIT無しの命令も二つの種類**があるからなのです。そして**第2－3表**の中にはその条件を満たす命令がないのです。それを次にみていくことに致しましょう。

《第2－15図》キー入力WAITあり

INKEY$の実験

仮にキー・スキャン用として“INKEY$”を用いたとします。そこでためしに**リスト2－10**のプログラムをキー・インして走らせてください。

INKEY$は，キーの入力があろうとなかろうと待ちませんから，キーの入力がないときでも次の命令に進みます。したがってキーに手を触れないときは，このプログラムはただ10行を高速でループしているだけです。

それでは“4”のキーを押し続けたらどうなるでしょうか？　INKEY$は“4”が押されたのをキャッチし，PRINT文で“4”をCRTに出力します。続いて次のマルチ・ステートメントによりまた10行に戻ります。するとまたINKEY$が“4”をキャッチして“4”を出力し——キーを押している間，これを繰り返すはずです（第2－16図－①）。

実際にゲームに利用するには，INKEY$により**キーの押されている間だけ**ラケットを動かせばよいのです——ところが。

ためしに“4”のキーを押してみてください。結果は見事にうらぎられました。10行でループしているにもかかわらず，なんと“4”は一つしかプリントされませんでした（第2－16図－②）。

もちろん“4”のキーをガチャガチャ何回も押せば

《リスト2－10》INKEY$の実験

```
10 PRINT INKEY$;:GOTO 10
```

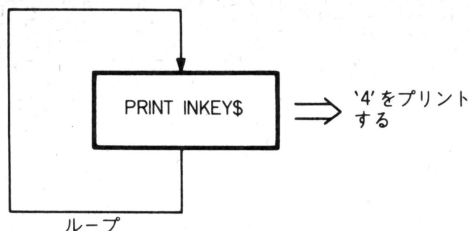
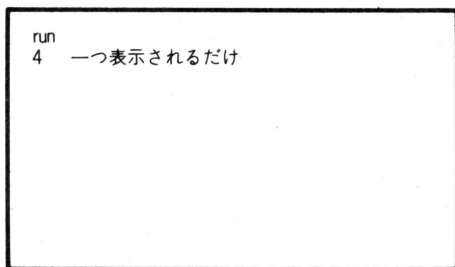
① リスト2－10の働き

PRINT INKEY$

'4'をプリントする

ループ

② 実際に'4'を押し続ける

run
4　一つ表示されるだけ

《第2－16図》INKEY$の実験

押した数だけ“4”がPRINTされますが，これではキーをガチャガチャ押さないとラケットが動かず，とてもゲームでは使いものになりません。

これは実はINKEY$が，キーの入力とともに以前に押されたキーをチェックしているからで，マシン語がおわかりの方はそのしくみもわかると思います。とにかく結論として，

INKEY$は

キー・スキャンには不向き！

ということがわかりました。それではどうしたら良いのでしょうか？

入力ポートをキャッチする

いろいろな方法がありますが，BASICでやるなら，

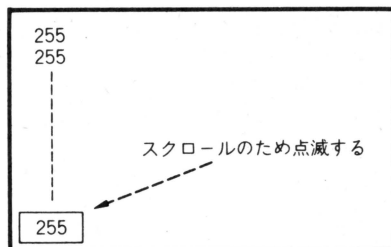
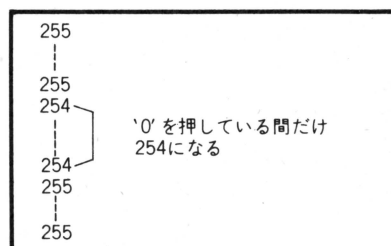
関数　INP（ポート番号）

を使うのがポピュラーでしょう。この関数はマシン語に近い――否，マシン語のINと同じもので，I/Oポートからの入力をキャッチする関数です。

というと難しくみえますが，簡単ですから実験してみましょう。

リスト2－11を走らせてみてください。すると**第2－17図－①**のようになるでしょう。これがキーの押されていない状態です。次にテン・キーの“0”を押し

《リスト2－11》INPの実験

```
10 PRINT INP(0):GOTO 10
```

《第2－17図》INP(0)の実験

たり，離したりしてみてください。どうです？　いちおう**第2－17図**の①～⑥の実験をしてみてください。

どうやらINPがキー・スキャンに使えそうなのがわかってきました。あとは255やら，248やらの意味さえわかれば良いのです。それを次に説明しますので，良く読んでください。

データ・バスの値を調べる

まず**ポート番号**について。

「PC－8001　USER'S　MANUAL」のP.77を見てください。**第2－18図**に同じものが載せてあります。この左の方に書いてある00～09の（インプットアドレス）というのがポート番号です。

（例）調べたいキー　　ポート番号

3　　　　0（00は0と同じ）

6

A チ　　　　2

だからテンキーの4が押されているか調べたければ，

INP（0）

の値をみれば良いのです。

ポート番号はOKですね。次にその返される値について。これには少し**2進数，16進数の知識**が必要ですから，苦手な人は良く読んでください。なにしろマシン語と同じ命令なのですから。

図の上の方にあるD0～D7（データ・バス）が入力データの各ビットを表わしていて，キーが押されていないときは1が立っています。すなわち，

1111　1111（データ・バス）
↓
FF　　　（16進数に変換）
↓
255　　　（10進数に変換）

先ほどの**第2－17図**－①の255は，このデータ・バスの値を10進数に変換したものだったのです。

次にキー4を押したとすると，D4の値が0になり，

D7D6D5D4D D3D2D1D0（図と順序が逆）
1110　1111
↓
EF　　　（16進数）
↓
239　　　（10進数）

になります。

以上のようにキーが押されたかどうかを調べるには，該当するキーのインプット・アドレスを指定して，関数INPによりデータ・バスの値を調べれば良いのです。

次にキー・スキャンの知識を利用して，実際にラケットを動かしてみることに致しましょう。

ラケットを動かす

（インプットアドレス）	D0	D1	D2	D3	D4	D5	D6	D7
00	0	1	2	3	4	5	6	7
01	8	9	*	+	=	,	.	RETURN
02	@ 〃	A チ	B コ	C ソ	D シ	E イ	F ハ	G キ
03	H ク	I ニ	J マ	K ノ	L リ	M モ	N ミ	O ラ
04	P セ	Q タ	R ス	S ト	T カ	U ナ	V ヒ	W テ
05	X サ	Y ン	Z ッ ツ	[「 。	¥ _	] 」 ム	^ ヘ	− = ホ
06	0 ヲ ワ	1 ! ヌ	2 〃 フ	3 # ア ア	4 $ ウ ワ	5 % エ エ	6 & オ オ	7 ' ヤ ヤ
07	8 (ュ ユ	9) ョ ヨ	: * ケ	; + レ	, < 、 ネ	. > 。 ル	/ ? ・ メ	− ロ
08	HOME CLR	↓ ↑	← →	INS DEL	GRAPH	カナ	SHIFT	CTRL
09	STOP	f・1	f・2	f・3	f・4	f・5	SPACE	ESC

（データ・バス）

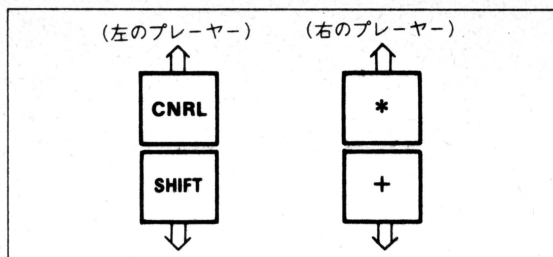
《第2－18図》データ・バス

まず，どのキーを押したらラケットが動くのかを決めましょう（**第2-19図**）。すると**第2-18図**から必要なインプット・アドレスとデータ・バスの値は**第2-20図**のようになります。そこで**第2－4表**のように各ポートの値を各変数に対応させます。

2100～2190のサブ・ルーチンで，キー・スキャンを行い，キーが押されていればキーによりラケットを動かします。まず2110でK1，K8にポート1，8の値を代入します。2120～2150でスキャンしています。たとえばCTRLキーが押されると，K8の値が127になりますから，2140により左のプレーヤーのラケットが一つ上に移動します。なお，

AND　LRA＞6

というのは，ラケットがコートの上端をつきやぶらないようにチェックしているのです。

さて実際のゲームの管理

《第2－19図》ゲーム用キーの配置

キーの種類	ポート番号	データの値
＋	1	247
*	1	251
CTRL	8	127
SHIFT	8	191

《第2－20図》ポート番号とデータ

をしているのは，2000からのサブルーチンで，**第2－21図**のようにループしているのがわかると思います。

どんなに時間がかかってもかまいません。**リスト2－12**を納得いくまで解析してみてください。

予告

第3章までで，ラケットを動かすことに成功しました。第4章は以上までの総まとめで，

GAMEを完成させる

《第2－4表》変数表のつづき

K1 K8	ポート1 〃 8 } の入力データの値

《第2－21図》2000番からのサブルーチンのループ

ところまで持っていきます。

ところで，ここまでのプログラムを走らせて，どうお感じになったでしょうか？

ラケットの動きが鈍い!

と思われたのではないでしょうか。悲しいかなこれは，「BASIC」の宿命で，このへんが**BASICの限界**なのです。そこで――。

一応BASICでGAMEが完成したあと，

マシン語の入門者を対象に

マシン語とのリンクをはかり

GAMEを**高速化**させる

予定です。御期待ください。

《リスト2－12》ラケットを動かすところまでのプログラム

```
10 REM ************************
20 REM *  ｶﾞﾝｿ TENNIS GAME      *
30 REM *       by K. ﾂｶｺﾞｼ      *
40 REM *         in FORESIGHT  *
50 REM *     <81.8.31-9.??>    *
60 REM ************************
99 '
100 REM ﾒｲﾝ ﾙｰﾁﾝ
110 WIDTH 40,25:CONSOLE 0,25,0,1:COLOR 7,0,0:PRINT CHR$(12)
120 GOSUB 1000 'ｼｮｷｾｯﾃｲ
130 GOSUB 1200 'ｼｮｷｾｯﾃｲ for RE-PLAY
140 GOSUB 1300 'ｶﾞﾒﾝ
150 GOSUB 1900 'ｻｰﾌﾞ
160 GOSUB 2000 'GAME ｦ ｵｺﾅｳ
998 GOTO 998
999 '
1000 REM ｼｮｷｾｯﾃｲ
1010 DIM TEN$(7,4),XMOV(4),YMOV(4)
1030 FOR I=0 TO 7
1031   FOR J=0 TO 4
1032     READ TEN$(I,J)
1033   NEXT J
1034 NEXT I
1040 READ SEN$
1050 FOR I=1 TO 4
1052   READ XMOV(I),YMOV(I)
1054 NEXT I
1090 RETURN
1099 '
1100 REM ﾃﾝ ﾉ ﾋｮｳｼﾞ
1110 X=0:IF LR$="r" THEN X=34
```

```
1120 FOR I=0 TO 4
1130   LOCATE X,Y+I:PRINT TEN$(TEN,I)
1140 NEXT I
1190 RETURN
1199 '
1200 REM ｼｮｷｾｯﾃｲ for RE-PLAY
1210 LTEN=0:RTEN=0:LRA=14:RRA=14
1290 RETURN
1299 '
1300 REM ｶﾞﾒﾝ
1310 PRINT CHR$(12)
1320 COLOR 6:LR$="l":TEN=LTEN:GOSUB 1100 'ﾋﾀﾞﾘ PLAYER ﾄｸﾃﾝ
1325         LR$="r":TEN=RTEN:GOSUB 1100 'ﾐｷﾞ  PLAYER ﾄｸﾃﾝ
1330 COLOR 1:LOCATE 0,5:PRINT SEN$;
1335         LOCATE 0,24:PRINT SEN$;
1340 COLOR 2:LOCATE 9,1:PRINT "*******************";
1342 COLOR 7:LOCATE 9,2:PRINT "    T E N N I S    ";
1344 COLOR 2:LOCATE 9,3:PRINT "*******************";
1350 GOSUB 1400 'ﾋﾀﾞﾘ ﾗｹｯﾄ
1360 GOSUB 1500  'ﾐｷﾞ ﾗｹｯﾄ
1390 RETURN
1399 ;
1400 REM ﾋﾀﾞﾘ ﾗｹｯﾄ
1410 COLOR 3
1420 LOCATE 0,LRA  :PRINT "■";
1430 LOCATE 0,LRA+1:PRINT "■";
1440 LOCATE 0,LRA+2:PRINT "■";
1490 RETURN
1499 ;
1500 REM ﾐｷﾞ ﾗｹｯﾄ
1510 COLOR 3
1520 LOCATE 37,RRA  :PRINT "■";
1530 LOCATE 37,RRA+1:PRINT "■";
1540 LOCATE 37,RRA+2:PRINT "■";
1590 RETURN
1599 ;
1600 REM BALL ｦ ｳｺﾞｶｽ
1610 XNXT=XBLL+XMOV(CRS):YNXT=YBLL+YMOV(CRS)
1620 IF XNXT<1  THEN GOSUB 1800:RETURN
1630 IF XNXT>36 THEN GOSUB 1800:RETURN
1640 IF YNXT<6  THEN GOSUB 1700:RETURN
1650 IF YNXT>23 THEN GOSUB 1700:RETURN
1660 COLOR 7:LOCATE XBLL,YBLL:PRINT "･"
1670 XBLL=XNXT:YBLL=YNXT
1680 LOCATE XBLL,YBLL:PRINT "●";
1690 RETURN
1699 '
1700 REM ｺｰﾄ
1710 ON CRS GOTO 1720,1730,1740,1750
1720 CRS=4:RETURN
1730 CRS=3:RETURN
1740 CRS=2:RETURN
1750 CRS=1:RETURN
1799 '
1800 REM ﾗｹｯﾄ
1810 ON CRS GOTO 1820,1830,1840,1850
1820 CRS=2:RETURN
1830 CRS=1:RETURN
1840 CRS=4:RETURN
1850 CRS=3:RETURN
1899 '
1900 REM ｻｰﾌﾞ
1910 COLOR 2:LOCATE 7,13:PRINT "◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆";
1920 COLOR 4:LOCATE 7,14:PRINT "     ｻ - ﾌﾞ";
1930 COLOR 7:LOCATE 7,15:PRINT "      Hit SPACE Key !";
1940 COLOR 2:LOCATE 7,16:PRINT "◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆";
1950 IF INKEY$<>" " THEN I=RND(1):GOTO  1950
1960 LINE (7,13)-(29,16)," ",7,BF
1970 CRS=INT(RND(1)*4)+1
1980 XBLL=18:YBLL=INT(RND(1)*12)+9:COLOR 7:LOCATE XBLL,YBLL:PRINT "●";;
1990 RETURN
```

```
1999 '
2000 REM GAME ヲ オコナウ
2010 GOSUB 1600 'ボール ヲ ウゴカス
2020 GOSUB 2100 'ラケット ヲ ウゴカス
2090 GOTO 2010
2099 '
2100 REM ラケット ヲ ウゴカス
2110 K1=INP(1):K8=INP(8)
2120 IF K1=251 AND RRA>6  THEN GOSUB 2300:RRA=RRA-1:GOSUB 1500
2130 IF K1=247 AND RRA<21 THEN GOSUB 2300:RRA=RRA+1:GOSUB 1500
2140 IF K8=127 AND LRA>6  THEN GOSUB 2400:LRA=LRA-1:GOSUB 1400
2150 IF K8=191 AND LRA<21 THEN GOSUB 2400:LRA=LRA+1:GOSUB 1400
2190 RETURN
2199 '
2300 REM ミギ ラケット ケス
2310 COLOR 3
2320 LOCATE 37,RRA  :PRINT " ";
2330 LOCATE 37,RRA+1:PRINT " ";
2340 LOCATE 37,RRA+2:PRINT " ";
2390 RETURN
2399 ;
2400 REM ヒダリ ラケット ケス
2410 COLOR 3
2420 LOCATE 0,LRA  :PRINT " ";
2430 LOCATE 0,LRA+1:PRINT " ";
2440 LOCATE 0,LRA+2:PRINT " ";
2490 RETURN
2499 ;
9000 DATA "◢■■◣"
9010 DATA "■  ■"
9020 DATA "■  ■"
9030 DATA "■  ■"
9040 DATA "◥■■◤"
9050 DATA " ◢■ "
9060 DATA "  ■ "
9070 DATA "  ■ "
9080 DATA "  ■ "
9090 DATA " ◢■◣"
9100 DATA "◢■■◣"
9110 DATA "■  ■"
9120 DATA "  ◢◤"
9130 DATA " ◢◤ "
9140 DATA "◢■■■"
9150 DATA "◢■■◣"
9160 DATA "■  ◤"
9170 DATA "  ■ "
9180 DATA "■  ◣"
9190 DATA "◥■■◤"
9200 DATA "◢   "
9210 DATA "■ ■ "
9220 DATA "■ ■ "
9230 DATA "◥■■◣"
9240 DATA "  ■ "
9250 DATA "■■■◣"
9260 DATA "■   "
9270 DATA "◥■■◣"
9280 DATA "   ■"
9290 DATA "◥■■◤"
9300 DATA "◢■■◣"
9310 DATA "■   "
9320 DATA "■■■◣"
9330 DATA "■  ■"
9340 DATA "◥■■◤"
9350 DATA "◢■■■"
9360 DATA "◤  ■"
9370 DATA "   ■"
9380 DATA "  ◢◤"
9390 DATA " ◢◤ "
9400 DATA "■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■"
9410 DATA 1,-1,-1,-1,-1,1,1,1:' BALL ノ ヘンイ
```

第4章 元祖「テニス・ゲーム」に挑戦

テニスゲーム完成！

はじめに

3章にわたって，テレビ・ゲームの元祖「テニス・ゲーム」を，少しずつ作ってきました。

第1章――画面を作る
第2章――ボールを動かす
(絵の動かし方)
第3章――ラケットを動かす
(キー・スキャンの原理)

のように一つずつ問題点を解決してきたわけです。これで必要なテクニックはすべて揃いました。そこでいよいよ第4章はそれらを総合して，

ゲームを完成させる！

ことに致しましょう。

遊び方

リスト2－13を御覧ください。これが第3章までのリストに改訂・追加をして，完成したプログラムです。これを順に説明していくわけですが，最初に遊び方をまとめておきましょう。

RUNさせると，**写真5**のようにGAMEエリアと両者の得点（0対0）が表示され，サーブ待ちとなります。スペース・キーを押すとゲーム開始となり，ボールがサーブされます。左右のプレーヤーは**第2－22図**のキーにより，ラケットを動かしてボールを受けるわけです（**写真6**）。もしボールをうけそこなうと，相手のポイントとなり，相手の得点が加算された上で，再びサーブ待ちとなります（**写真7**）。

こうしてゲームを続け，どちらかのプレヤーが**7点**

《写真5》RUN直後の画面

《写真6》ゲーム開始

《写真7》レシーブミス（再びサーブ待ち）

《写真8》GAME OVER

《第2－5表》サブルーチンの内容

1000	第1初期設定
1100	得点の表示 （入力）TEN←表示したい得点 LR $← { l……左のプレーヤーの得点時 r……右　〃
1200	第2初期設定
1300	固定画面の出力
1400	左のプレーヤーのラケット表示 （入力）LRA←ラケットの上端の座標
1500	右のプレーヤーのラケット表示 （入力）RRA←ラケット上端の座標
1600	ボールを動かす
1700	コートに当たったとき，ボールを反射させる
1800	ボールが右端にきたときの処理 （出力）GAME＝ { 0：ボールを受けたとき 1：受けそこねたとき 2：受けそこね，さらに相手の得点＝7になったとき
1900	サーブをおこなう （ボールの初期位置とコースを決める）
2000	GAMEをおこなう （出力）1800のルーチンと同じ
2100	キー・スキャン キーが押されていれば，ラケットを動かす
2200	ボールが左端にきたときの処理 （出力）1800のルーチンと同じ
2300	右のラケットを消す
2400	左のラケットを消す
2500	RE－RLAY 処理 （出力）I $＝ { "1"……再GAMEするとき "2"……　〃　しないとき
2600	タイマー

を取るとＧＡＭＥ終了となります（**写真8**）。この状態で“1”のキーを押せば，再ゲーム可能です。また“2”のキーを押せば，ＢＡＳＩＣのコマンド待ちとなるわけです。

《第2－22図》ゲーム用キーの配置

メイン・ルーチン

いかがですか？

第4章は，以上のところまで持っていきたいと思います。それでは最初に**フローチャート1**を見てください。

100～999がメイン・ルーチンです。

ここは第3章に比べ，当然変更してあります。まず110で画面をクリアし，画面モードの設定をおこないます。次の初期設定というのは，**プログラムで必要な変数の値を初期化**するものです。「第1初期設定」で全体に関係のある変数を，「第2初期設定」では再ゲームの度に必要な変数の設定をします。

140で画面を作り，150でサーブをおこないます。160で実際のゲームが進行します。この160でCall しているサブ・ルーチンの中でＧＡＭＥという変数を導入しました。この変数はメイン・ルーチンを管理する上で重要ですから，少し説明を加えておきましょう。

変数GAMEの役割

リストの**1970行**をみてください。第3章のリストに

《フローチャート1》メイン・ルーチン

数字は行番号
(　)内はサブ・ルーチンの行番号

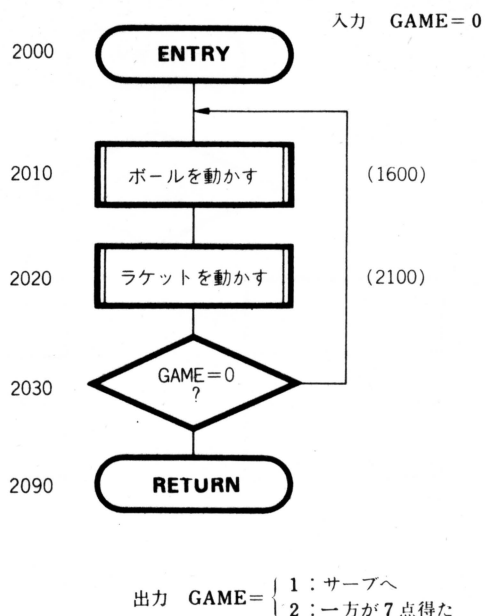
《フローチャート2》GAMEをおこなう

出力　GAME＝{ 1：サーブへ
2：一方が7点得た

GAME＝0という式を追加しました。ということは，1900のサーブのサブルーチに飛び込む度に，変数GAMEの値は0にクリアされます。すると当然160行でGAMEをおこなうサブ・ルーチンに入る時は，変数GAME＝0になっています。

そこで**フローチャート2**を御覧ください。

変数GAME＝0の間中，ボールとラケットを交互に動かしているのがわかるでしょう。するとどういうとき，変数GAMEの値が変わるのでしょうか？

1840行をみてください。ここで変数GAMEの値が変化しています。これは右のプレーヤーがボールを受けそこなったところです。まずGAME＝1としたところで**IF文**により，相手の得点＝7（つまりGAME　OVER）になっているかチェックしています。つまり，

　ボールを受けそこなう → GAME＝1
　さらに相手の得点＝7 → GAME＝2

にしているのです。同様に左プレーヤーについても**2240行**でチェックしています。

こうしてGAME≠0になったところで，2000のサブ・ルーチンを抜けて，再び**フローチャート1**の170行に戻ります。ここでGAME＝1なら，サーブへ。またGAME＝2なら，GAME　OVERですから，180のRE－PLAY処理に移ります。

2500のサブ・ルーチンでは，

　再ゲームをする――I＄＝“1”
　ゲームをやめる――I＄＝“2”

で戻ってきますから，それを190行でチェックし，再ゲームさせたり（130行），BASICに戻したりして（999行）プログラムが終了します。

ラケットの反射をキャッチする

以上でプログラム全体の流れがおわかりいただけたと思います。次に，第3章からの続きとしていよいよ**最後の問題点**である。

　ボールとラケットの反射

をどう処理するかをみていくことに致しましょう。

《フローチャート3》右のラケット・チェック

1800 ENTRY

1810 スティックで受けた？

Yes

No

音を出す (2600)

ボールのコース変える

左のプレーヤーの得点加算

左プレーヤーの得点表示 (1100)

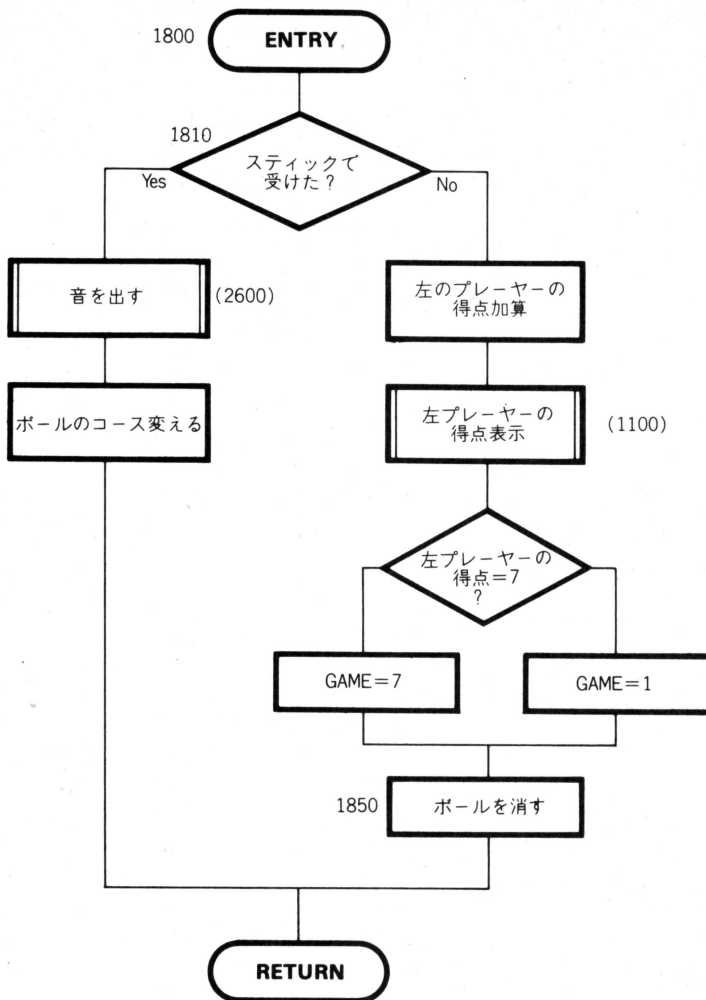
左プレーヤーの得点＝7？

GAME＝7

GAME＝1

1850 ボールを消す

RETURN

《第2－23図》ラケットとボールの反射をチェックする

1620が左のプレーヤー，1630が右のプレーヤーのチェックです。これを2200と1800のサブ・ルーチンに分けて処理することにしました。そこで1800の右プレーヤーのラケットについてみてみることにします。左プレーヤーについても同様です。

フローチャート3がその処理の流れ図です。

ここで問題になるのは，1810の判断のところでしょう。そこで**第2－23図**を見てください。

ＲＲＡがラケットの上端のＹ座標，ＹＢＬＬがボールのＹ座標ですからこれらを比較して，

ＲＲＡ≦ＹＢＬＬ≦ＲＲＡ＋２

であれば，うまくラケットにボールを当てたことになります。**第2－23図**の式をまとめたのが1810行ですが，**第2－23図**の式の方がわかり良ければ，そちらの式を用いてください。

なお1850行でボールを消しているのは，再サーブに移ったとき，前のボールが消えずに残ってしまうからです。

その他の注意点

① 1110行にＣＯＬＯＲ 6（黄色）を追加してください。これがないと，得点を変更した時に白で表示されてしまいます。

② 1660行を"　"に変えてください。これでボー

ルの跡が残らなくなります。

③ データ文9350〜9390の得点7の表示が評判が悪かったので，リストのように変更しました。

仕上げ—効果音をつける

以上でGAMEの製作はすべて終りました。最後に仕上げとして，効果音を入れます。

PC－8001では**BEEP**しかありませんが，これだけでも実にさまざまな音が出せます。その音色はまさに無限といってよく，どんな音を取り入れるかは，

各自のアイデア次第

というわけです。効果音がGAMEに与える影響はなかなかなもので，決して馬鹿に出来ません。

GAMEの出来映えは，効果音で決まる！

と言って良く，ゆめゆめおろそかにすることなかれ。

第4章ではGAME作りへの入門というわけで，音はボールがコートに当たったときと，ラケットに当たったときに出しています。音を出すところは，2600〜2690にまとめてあり，2610からCallすれば「ピピッ」，2630からCallすれば，「ピッ」と鳴ります。

したがって皆さんはこのサブ・ルーチンを，自分のくふうでいろいろに変えてみてください。マシン語を使えば，音楽を奏でることさえ可能です。

M子ちゃんとの対話

ツカ：M子ちゃんは，マイコンをどんな風に使っているの？

M子：たまに人の作ったGAMEをするくらい。

ツカ：つまり，初心者というワケですね。それで，「GAMINGへの招待」を4章とも読んでもらったんだけど，どうでした？

M子：——ウーン。

ツカ：難しかった？

M子：難しいのかやさしいのかわからない。だって私，BASIC知らないんだもん。

ツカ：——？　ハア，御苦労様でした（皆さんの中にはM子ちゃんはいなかったでしょうね）。

読者への助言（おわりに代えて）

今シリーズはある程度BASICの文法をマスターした人を対象に，「テニス・ゲーム」を題材にリアル・タイム・ゲームの作り方を見てきました。

もし難しいと思われた方は，まだBASICの文法の理解が不足です。ここに現われたキー・ワードの苦手なものを調べなおしてから，何度も挑戦してみてください。

リアル・タイム・ゲームの作り方が理解出来た人——結構ですね。今度は別のプログラムに挑戦してみてください。そして完成したら，ぜひ「マイコン誌」で御披露してください。

最後に内容がやさしすぎた人。逆に何か良い教材がありましたら，御教授願います。

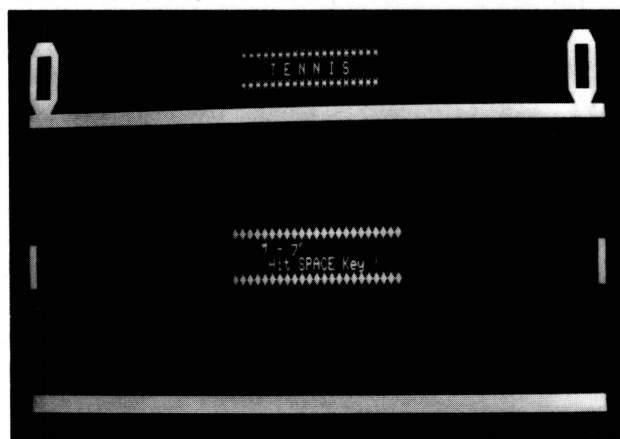
第5章はテニスシリーズの番外編として，前章でも予告したように，マシン語の入門者を対象に，本プログラムとマシン語とのリンクを考えてみたいと思います。御期待ください。

《リスト2－13》テニスゲーム完成版

```
10 REM ************************
20 REM *  ｶﾞﾝﾂ TENNIS GAME    *
30 REM *       by K. ﾂｶｺﾞｼ     *
40 REM *        in FORESIGHT  *
50 REM *    <81.8.31-9.15>    *
60 REM ************************
99 '
100 REM ﾒｲﾝ ﾙｰﾁﾝ
110 WIDTH 40,25:CONSOLE 0,25,0,1:COLOR 7,0,0:PRINT CHR$(12)
120 GOSUB 1000 'ｼｮｷｾｯﾃｲ
130 GOSUB 1200 'ｼｮｷｾｯﾃｲ for RE-PLAY
140 GOSUB 1300 'ｶﾞﾒﾝ
150 GOSUB 1900 'ｻｰﾌﾞ
160 GOSUB 2000 'GAME ｦ ｵｺﾅｳ
170 IF GAME=1 THEN 150
180 GOSUB 2500 'replay ｼｮﾘ
190 IF I$="1" THEN 130
998 END
999 '
1000 REM ｼｮｷｾｯﾃｲ
1010 DIM TEN$(7,4),XMOV(4),YMOV(4)
1030 FOR I=0 TO 7
1031   FOR J=0 TO 4
1032     READ TEN$(I,J)
1033   NEXT J
1034 NEXT I
```

```
1040 READ SEN$
1050 FOR I=1 TO 4
1052   READ XMOV(I),YMOV(I)
1054 NEXT I
1090 RETURN
1099 '
1100 REM ﾃﾝ ﾉ ﾋｮｳｼﾞ
1110 COLOR 6:X=0:IF LR$="r" THEN X=34
1120 FOR I=0 TO 4
1130   LOCATE X,Y+I:PRINT TEN$(TEN,I)
1140 NEXT I
1190 RETURN
1199 '
1200 REM ｼｮｷｾｯﾃｲ for RE-PLAY
1210 GAME=0:LTEN=0:RTEN=0:LRA=14:RRA=14
1290 RETURN
1299 '
1300 REM ｶﾞﾒﾝ
1310 PRINT CHR$(12)
1320 COLOR 6:LR$="l":TEN=LTEN:GOSUB 1100 'ﾋﾀﾞﾘ PLAYER ﾄｸﾃﾝ
1325         LR$="r":TEN=RTEN:GOSUB 1100 'ﾐｷﾞ  PLAYER ﾄｸﾃﾝ
1330 COLOR 1:LOCATE 0,5:PRINT SEN$;
1335        LOCATE 0,24:PRINT SEN$;
1340 COLOR 2:LOCATE 9,1:PRINT "**********************";
1342 COLOR 7:LOCATE 9,2:PRINT "    T E N N I S    ";
1344 COLOR 2:LOCATE 9,3:PRINT "**********************";
1350 GOSUB 1400 'ﾋﾀﾞﾘ ﾗｹｯﾄ
1360 GOSUB 1500  'ﾐｷﾞ ﾗｹｯﾄ
1390 RETURN
1399 ;
1400 REM ﾋﾀﾞﾘ ﾗｹｯﾄ
1410 COLOR 3
1420 LOCATE 0,LRA  :PRINT "■";
1430 LOCATE 0,LRA+1:PRINT "■";
1440 LOCATE 0,LRA+2:PRINT "■";
1490 RETURN
1499 ;
1500 REM ﾐｷﾞ ﾗｹｯﾄ
1510 COLOR 3
1520 LOCATE 37,RRA  :PRINT "■";
1530 LOCATE 37,RRA+1:PRINT "■";
1540 LOCATE 37,RRA+2:PRINT "■";
1590 RETURN
1599 ;
1600 REM BALL ｦ ｳｺﾞｶｽ
1610 XNXT=XBLL+XMOV(CRS):YNXT=YBLL+YMOV(CRS)
1620 IF XNXT<1  THEN GOSUB 2200:RETURN
1630 IF XNXT>36 THEN GOSUB 1800:RETURN
1640 IF YNXT<6  THEN GOSUB 1700:RETURN
1650 IF YNXT>23 THEN GOSUB 1700:RETURN
1660 COLOR 7:LOCATE XBLL,YBLL:PRINT " ";
1670 XBLL=XNXT:YBLL=YNXT
1680 LOCATE XBLL,YBLL:PRINT "●";
1690 RETURN
1699 '
1700 REM ｺｰﾄ
1710 GOSUB 2630:ON CRS GOTO 1720,1730,1740,1750
1720 CRS=4:RETURN
1730 CRS=3:RETURN
1740 CRS=2:RETURN
1750 CRS=1:RETURN
1799 '
1800 REM ﾐｷﾞ ﾗｹｯﾄ ﾁｪｯｸ
1810 IF (YBLL-RRA)*(YBLL-RRA-2)<=0 THEN GOSUB 2610:CRS=(CRS+5)/3:RETURN
1820 LTEN=LTEN+1 'ﾋﾀﾞﾘ ﾌﾟﾚｰﾔｰ ﾎﾟｲﾝﾄ
1830 TEN=LTEN:LR$="l":GOSUB 1100
1840 GAME=1:IF LTEN=7 THEN GAME=2
1850 LOCATE XBLL,YBLL:PRINT " "
1890 RETURN
1899 ;
1900 REM ｻｰﾌﾞ
1910 COLOR 2:LOCATE 7,13:PRINT "◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆";
1920 COLOR 4:LOCATE 7,14:PRINT "     ｻ - ﾌﾞ";
1930 COLOR 7:LOCATE 7,15:PRINT "      Hit SPACE Key !";
1940 COLOR 2:LOCATE 7,16:PRINT "◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆";
1950 IF INKEY$<>" " THEN I=RND(1):GOTO  1950
1960 LINE (7,13)-(29,16)," ",7,BF
1970 GAME=0:CRS=INT(RND(1)*4)+1
1980 XBLL=18:YBLL=INT(RND(1)*12)+9:COLOR 7:LOCATE XBLL,YBLL:PRINT "●";;
1990 RETURN
2000 REM GAME ｦ ｵｺﾅｳ
2010 GOSUB 1600 'ﾎﾞｰﾙ ｦ ｳｺﾞｶｽ
2020 GOSUB 2100 'ﾗｹｯﾄ ｦ ｳｺﾞｶｽ
2030 IF GAME=0 THEN GOTO 2010
2090 RETURN
2099 '
2100 REM ﾗｹｯﾄ ｦ ｳｺﾞｶｽ
```

```
2110 K1=INP(1):K8=INP(8)
2120 IF K1=251 AND RRA>6  THEN GOSUB 2300:RRA=RRA-1:GOSUB 1500
2130 IF K1=247 AND RRA<21 THEN GOSUB 2300:RRA=RRA+1:GOSUB 1500
2140 IF K8=127 AND LRA>6  THEN GOSUB 2400:LRA=LRA-1:GOSUB 1400
2150 IF K8=191 AND LRA<21 THEN GOSUB 2400:LRA=LRA+1:GOSUB 1400
2190 RETURN
2199 '
2200 REM ﾋﾀﾞﾘ ﾗｹｯﾄ ﾁｪｯｸ
2210 IF (YBLL-LRA)*(YBLL-LRA-2)<=0 THEN GOSUB 2610:CRS=CRS*3-5:RETURN
2220 RTEN=RTEN+1 'ﾐｷﾞ ﾌﾟﾚｰﾔｰ ﾎﾟｲﾝﾄ
2230 TEN=RTEN:LR$="r":GOSUB 1100
2240 GAME=1:IF RTEN=7 THEN GAME=2
2250 LOCATE XBLL,YBLL:PRINT " "
2290 RETURN
2299 ;
2300 REM ﾐｷﾞ ﾗｹｯﾄ ｹｽ
2310 COLOR 3
2320 LOCATE 37,RRA  :PRINT " ";
2330 LOCATE 37,RRA+1:PRINT " ";
2340 LOCATE 37,RRA+2:PRINT " ";
2390 RETURN
2399 ;
2400 REM ﾋﾀﾞﾘ ﾗｹｯﾄ ｹｽ
2410 COLOR 3
2420 LOCATE 0,LRA  :PRINT " ";
2430 LOCATE 0,LRA+1:PRINT " ";
2440 LOCATE 0,LRA+2:PRINT " ";
2490 RETURN
2499 ;
2500 REM ｹﾞｰﾑ ｵｰﾊﾞｰ
2510 COLOR 5:LOCATE 7,13:PRINT "***********************";
2520 COLOR 2:LOCATE 7,14:PRINT "  G A M E   O V E R !  ";
2530 COLOR 7:LOCATE 7,15:PRINT "     1･･･r e p l a y   ";
2540         LOCATE 7,16:PRINT "     2･･･s t o p       ";
2550 COLOR 5:LOCATE 7,17:PRINT "***********************";
2560 I$=INKEY$:IF I$<>"1" AND I$<>"2" THEN 2560
2570 LINE (7,13)-(29,17)," ",7,BF
2590 RETURN
2599 '
2600 REM ｵﾄ
2610 BEEP 1:FOR TT=0 TO 40:BEEP 0:NEXT
2620        FOR TT=0 TO 20:NEXT
2630 BEEP 1:FOR TT=0 TO 50:BEEP 0:NEXT
2690 RETURN
2699 '
9000 DATA "◢■■◣"
9010 DATA "■  ■"
9020 DATA "■  ■"
9030 DATA "■  ■"
9040 DATA "◥■■◤"
9050 DATA " ◢■ "
9060 DATA "  ■ "
9070 DATA "  ■ "
9080 DATA "  ■ "
9090 DATA " ◢■◣"
9100 DATA "◢■■◣"
9110 DATA "■  ■"
9120 DATA "  ◢◤"
9130 DATA " ◢◤ "
9140 DATA "◢■■■"
9150 DATA "◢■■◣"
9160 DATA "■  ◤"
9170 DATA "  ■ "
9180 DATA "■  ◣"
9190 DATA "◥■■◤"
9200 DATA "◢   "
9210 DATA "■ ■ "
9220 DATA "■ ■ "
9230 DATA "◥■■◣"
9240 DATA "  ■ "
9250 DATA "■■■◣"
9260 DATA "■   "
9270 DATA "◥■■◣"
9280 DATA "   ■"
9290 DATA "◥■■◤"
9300 DATA "◢■■◣"
9310 DATA "■   "
9320 DATA "■■■◣"
9330 DATA "■  ■"
9340 DATA "◥■■◤"
9350 DATA "■■■■"
9360 DATA "■  ■"
9370 DATA "   ■"
9380 DATA "  ◢◤"
9390 DATA " ◢◤ "
9400 DATA "■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■"
9410 DATA 1,-1,-1,-1,-1,1,1,1:' BALL ﾉ ﾍﾝｲ
```

第5章 元祖「テニス・ゲーム」に挑戦
（番外編）マシン語による高速化

はじめに

元祖「テニス・ゲーム　に挑戦！”
編も第5章となりました。そして番外編のサブ・タイトルは

マシン語による高速化

です。カッコいいですね，凄いですね，シビアですね。

マシン語を知らないあなた！

BASICしか知らないあなた！

ゲームの好きなあなた！

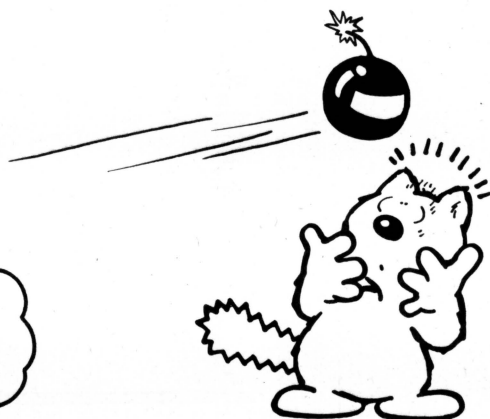
ゲームの嫌いなあなた！

結構，結構。寄ってらっしゃい，見てらっしゃい。
マシン語編のはじまり，はじまり。ジャジャジャ〰〰ン。

《第2－6表》シリーズの内容

章	目　　標
第1章	ゲームの画面を作る
第2章	絵を動かす原理を知る（ボールを動かす）
第3章	キースキャンの原理を知る（ラケットを動かす）
第4章	仕上げ，ゲームの完成

第5章の目標

第4章でお約束したように，第5章ではBASIC版の「テニス・ゲーム」にマシン語をリンクさせ，若干の

高速化を試みようと思います。その方針としては，

① 第4章のリストを尊重し，キーインの手間をできるだけ省けるよう，**変更点を最少**にすること。

② 追加するマシン語も出来るだけ少なくする。

③ マシン語の予備知識を出来るだけ最少に押え，多くの人が参加出来るようにする。

です。したがってマシン語の知識としては，川村　清さんの手による月刊マイコン連載の「マシン語講座　基礎編」をお読みの方なら，十分過ぎるくらい十分です。追加したマシン語は数10バイトにすぎませんし，XOR命令を使えば，出力・消去ルーチンを共通化出来ますのでほぼ半分に減らせます。さらにDAD命令（インテル表記）を使えば，もう数バイト減らせます。

しかしここでは入門ということもあり，出来るだけわかりやすい方法を取りました。

なお不幸にしてマシン語音痴の人がいましたら，ゲームの入力から遊び方までを読み，「今後の改良点」までを読み飛ばしてください。ゲームの作り方は第4章までで終っていますから今後BASICのみによる改良を続けて行けば良いでしょう。なおその場合でも真ん中でUSR関数の説明をしてありますので，いつか読み直していただくと参考になるかもしれません。

プログラムの入力

プログラムの入力は，次のようにおこなってください。

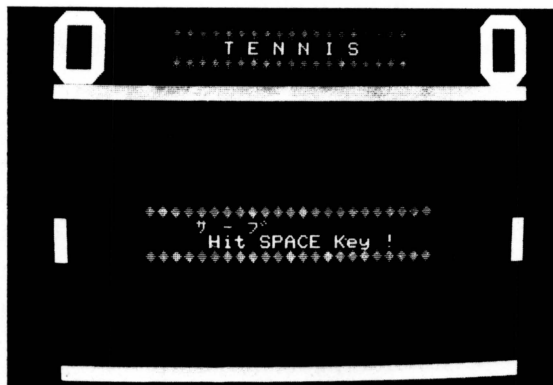
1．マシン語プログラムの入力

モニタを使って入力すること

MON↗　　＊SD000↗

アドレスは，

D000～D053番地

です。このうち“NOP”と書いてあるところは入力不要です。

《写真9》ゲーム・スタート

2．BASIC部

第4章までのリストを入力されていない方は，御苦労様ですがリストをすべて打ち込んでください。

第4章までのオールBASIC版を完成している人は，下の変更・追加の部分のみで結構です。

① マシン語の設定部分（追加）
105～109行

② 変数の整数化――115行を追加

③ ラケット部のアトリビュート設定（追加）
1345～1346行

④ ボール・ラケットの入出力（変更）

1660，1680行
1400
1500　のサブルーチン
2300
2400
リストのように変更

遊び方

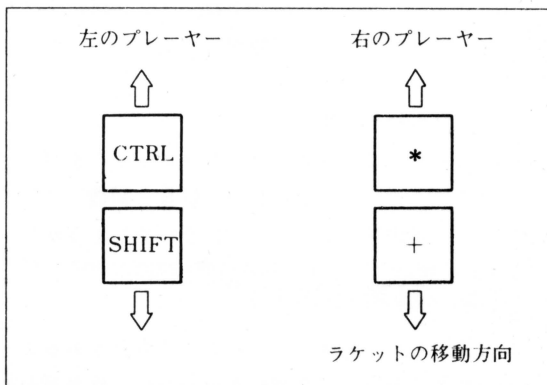
オールBASIC版とまったく同じです。いちおう初めての人のために説明しておきます。

プログラムのスタートはBASICモード（BASICのコマンドレベル）で

RUN↗

です。**写真9**がゲーム・スタート時の状態です。

このゲームは最近のマイコン・ゲームでは珍しい2人用ゲームです（だから**元祖テニス・ゲーム**です）。ですからこの状態のとき，**第2－24図**のキー配置を見て，2人とも指のスタンバイしてください（不幸にも相手がいない時は，左手と右手で勝負しましょう）。画面の上部左右に2人の得点が表示されています。

SPACEキーを押すと，ボールがサーブされます（**写真10**）。それぞれ自分のラケットを操作して，ボールを打ち返してください。ボールを受けそこなうと，

《第2－24図》ゲームのキー操作

《写真10》サーブ・ボール

《写真11》ゲーム・オーバー

相手の得点となります(得点は自動的に表示されます)。

こうしてどちらかが7点を取るとゲーム終了となります。**写真11**がその状態を表わしています。この時再ゲームがしたければ′1′を，ゲームをやめたければ′2′を押してください。BASICのコマンド・レベルに戻り，プログラム終了となります。

USR関数

さてこれからマシン語とのリンクを考えることにします。

普通BASICとマシン語とのリンクをはかるときは，

① BASICでは出来ない部分
② 高速化をはかりたい部分

のみをマシン語で作り（マシン語のサブルーチン）それをメインルーチンであるBASICから呼び出して使う方法をとります。

BASICからマシン語のサブルーチンを呼ぶときは，

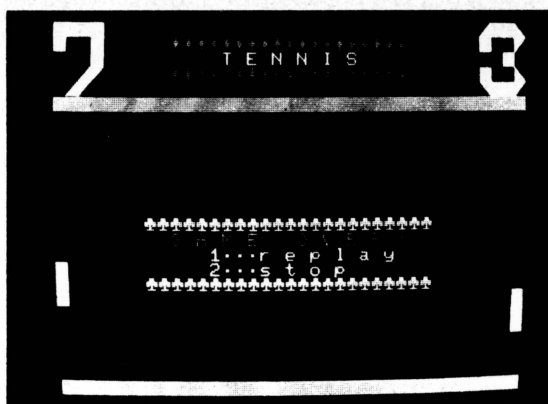
GOSUB

ではダメです。マシン語のプログラムには，行番号がありません。だから当り前ですね。マシン語のサブルーチンを呼ぶには専門の関数があります。それがいわゆる

USR関数

と呼ばれるものです。マイコンのマニュアルをパラパラと見ていくと，おそらくこの“**USR関数**”に関する項目が一番理解しずらいのではないでしょうか？

マシン語領域の設定

BASICには各種マシンによって方言があることは御存知ですね。USR関数もマシンにより方言があり——というよりはマシンによって使い方がマチマチです。そこで以下に説明するUSR関数の使い方は，PC—8001を主体とするものであることをお断りしておきます。

PC—8001のマニュアルにおけるUSR関数の項も圧縮して記述されているので，読みにくいかもしれません。以下に少しずつほぐしていきましょう。

「ユーザーのアセンブリ言語サブルーチンのためのメモリスペースは，それをロードする前に確保しておかなければなりません。使用できるメモリ領域は，図に示すRAM領域の最後部で，その大きさはCLEARコマンドの2番目のパラメータにより設定します。（『リファレンス・マニュアル』P. 97)

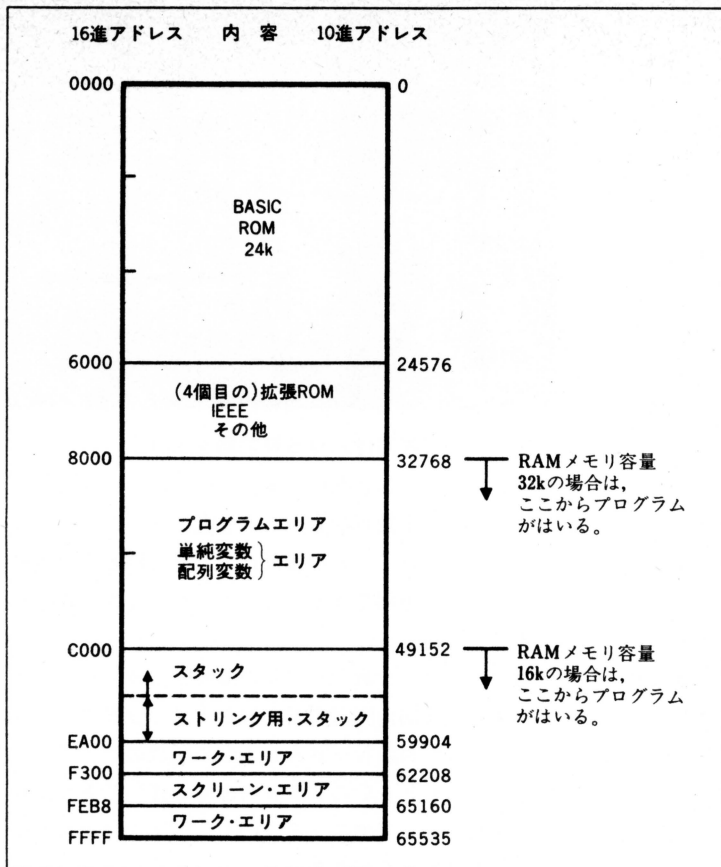
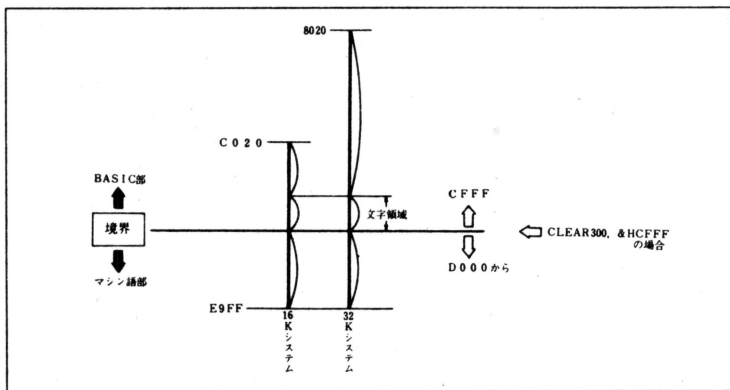
図とは**第2—25図**のことです。これだけを読んで何を言っているのかわかりますか？

電源ONの状態では，RAMのフリー・エリアすべてをBASICが使えるようになっています。このままの状態でマシン語のプログラムを入力しても，BASICのプログラムを走らせると破壊されてしまいます。そこでメモリのフリー・エリアを2分割し，

前半：BASIC
後半：マシン語

のように分けて使う必要があります。この時両者の境界線を定めるのがCLEAR文です。

第2—26図でその使い方を見てみましょう。

まずRAMのフリー・エリア（使用者が自由に使える範囲）の下限はE9FF番地までですが，上限は各自のRAM容量によって異なります。マニュアルの**第2図**によると32K，16Kシステムでそれぞれ8000番地，C000番地となっていますが，フリー・エリアの最初の32バイト（16進数で20番地分）は，ハード・リセットにより破壊されますから，**第2—26図**のようにフリー・エリアの上限は，

《2－25図》メモリマップ

《第2－26図》RAMのフリー・エリアとCLEAR文

32Kシステム──→8020番地

16Kシステム──→C020番地

とした方が安全でしょう。

第2－26図ではBASICの領域をCFFF番地までとしています。そこでこの値をCLEAR文の第2パラメータにセットし，

CLEAR　300，&HCFFF
第1パラメータ　　第2パラメータ

と宣言すれば，マシン語領域としてD000番地以降が使えます。なお蛇足ながら“&H”は16進数を表わしています。

CLEAR文には他に第1パラメータとして文字領域の大きさも設定出来ますが，今回の主旨とかけ離れますのでここでは省略します。

リスト2－14の105行が，マシン語の領域を確保しているところです。したがってD000番地からマシン語のプログラムを書けます。**リスト2－15**を見ると，確かにD000番地から書かれているのがわかります。

ユーザー関数の定義

次に**ユーザー関数の定義**です。

リスト2は，四つのサブルーチンから構成されています。そこでその四つに，異なるユーザー関数を割り当てます。ユーザー関数は全部で10個あり，

USR0
USR1
〜
USR9

のように0〜9の数で区別されています。ここでは，USR1〜USR4を次のように割り当てることにします。

USR1：D000番地からのサブルーチン
USR2：D010番地
USR3：D020番地
USR4：D040番地

USR関数の割り当てには，

DEFUSR

を用います。具体的には，**リスト2－14**の106〜109行を御覧ください。

これにより以下BASICの中で**D000番地からのサブルーチンは，USR1**という**関数に変化**しました。他のルーチンについても同様です。

ユーザー関数の引き数

さて次が，BASICとマシン語ルーチンとの引き数のやり取りの理解です。

例をあげましょう。**リスト2－15**のD000番地か

らはボールをPRINTするルーチンが書かれています。そのしくみを大まかに言えば，

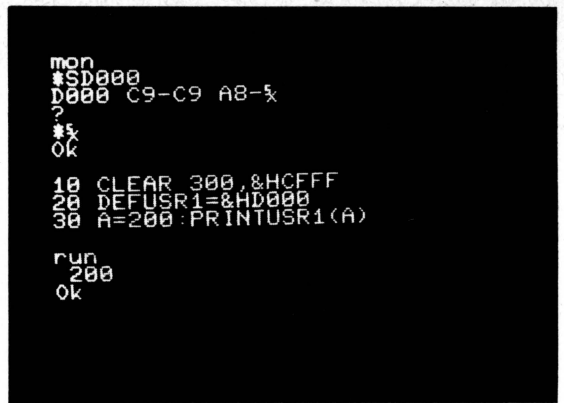
BASICでボールの位置を指定
⇩
マシン語ルーチンでボールを出力

という方法で，BASICとマシン語で

ボールの位置

というデータがやり取りされています。USR関数では引き数を使ってBASICとマシン語とのデータの引き渡しを行っています。その方法をマニュアルで調べようとすると，かなりシビアなのがわかります。そこでそれに挑戦することにします。

《写真12》USR関数をテスト

「われわれが，PC−8001 N−BASICの扱う浮動小数点表現について説明しうることは，この章に記述されていることだけです。……したがって，PC−8001の浮動小数点形式データを直接扱うユーザーは，その章に書かれている情報だけで，その浮動小数点形式のすべてを理解できるだけの力を持った人に限られます」

と書いてあり，変数テーブルの構成が掲げられています（**第2−28図**）。同図のVARPTRは変数のアドレスを調べるためのN−BASICの関数のことです。結局ここにもFACのアドレスまでは書いてありませんでした。そしてマニュアル曰く，同図を見てもわからない人はあきらめなさいと。あなたならどうしますか？　さてマニュアルによれば，わかる人ならわかるはずだと言うのだからFACのアドレスは求まるはずです。実はマニュアルをもっと良く読めば，簡単に求まるのです。リファレンス・マニュアルのP.98を見ると，「引き数が数値の場合には，〔HL〕レジスタペアには引き数が格納されている浮動小数点アキュームレータへのポインタ（FAC−3）のアドレスが格納されています」

と書かれています。これによりFAC−3のアドレスが求まります。

具体的には，まず私がすでに発表した「レジスタ表示プログラム」（「マイコン誌」82年1月）を入力し

「引き数が整数の場合には，

FAC−3　が引き数の下位8ビットを，

FAC−2　が引き数の上位8ビットを，

保持します」（同，P.98）

何のことかわかりますか？　FAC−2，FAC−3と言っても，マニュアルの前後を見ても出てきません。困りましたね。そこで，**第2−27図**のBASICとマシン語（わずか1バイト)を入力して走らせてみてください。**写真12**が実行したところです。30行でAに入っている200というデータを引き数としてマシン語に渡していますが，マシン語のサブでは何もしていないので（したがってデータも加工されていない)，200というデータがそのままBASICに戻され，PRINTされたものです。

マシン語ルーチンで，この引き数をどのように利用したら良いのでしょうか？

「マニュアル」へ挑戦

実は，変数表については「PC−8001のユーザーズ・マニュアル」の「変数テーブル」（P.58）に説明があります。ところがそこを読むと，

① BASICプログラム

```
10 CLEAR 300, &HCFFF
20 DEFUSR1=&HD000
30 A=200：PRINT USR1 (A)
```

② マシン語プログラム

アドレス	マシン語	アセンブリ言語
D000	C9	RET

《第2−27図》USR関数をテストする

《第2−28図》変数テーブルの構成

それが使えるようにシステム・ワークエリアのF1E3番地からの3バイトをセットしてください。そして第2—27図のマシン語のプログラムのD000番地を

FF——RST　38H

に変えた後，BASICモードに戻して

RUN↗

で走らせてみてください。第2—29図のように表示されるでしょう（なお同図においては，A，HL以外の汎用レジスタの値は不定です)。この時のHLレジスタの値　F0A8番地

がFAC−3のアドレスです。なぜならプログラムの制御がUSR関数に移り，マシン語サブルーチンの実行に移った時，HLレジスタはFAC−3を指すとマニュアルに書いてあり，第2—29図はまさにその時のレジスタの値を表示したものだからです。

以上で我々はUSR関数の使い方を完全にマスターしたわけです。どうですか？

「それは公表できる性質のものではありません」(「ユーザーズ・マニュアル」P.58)

の部分に関することですから，少々難しかったかもしれません。次節でボールの表示ルーチンを例に，もう少し具体的に説明してみましょう。

USR関数を使う

ユーザー関数で引き数を使う時は，引き数に整数を用いるのが便利です。リスト2−14の115行目はそのためのものです。こうすることにより，マシン語側は2バイトのデータとして引き数を受け取ることが出来ます。　1660行で変数XYにボールの座標を入れて，USR関数をCALLしています。すると

FAC−3（F0A8番地）：ボールタテ座標

FAC−2（F0A9番地）：ボールヨコ座標

が格納されますから，D000番地の

LD　HL，（0F0A8H)

により，

Hレジスタ←—ボールのヨコ座標

Lレジスタ←—ボールのタテ座標

を得ることが出来ました。

第5章ではBASICとマシン語をリンクさせるときのユーザー関数の使い方を理解していただくのが目的ですから，D005番地でCALLしているシステム・サブルーチンのみ解説しておき，これ以上マシン語の中身にタッチするのは避けることにします。

3F3番地からのサブルーチン

LOCATE座標を，実際のビデオRAMのアドレスに変換する。ただし入力の時LOCATE座標を1オリジンの16進数に変換し，

H←—ヨコ座標

L←—タテ座標

にセットしておく。結果はHLレジスタに得られる。

今後の改良点

以上をもちまして本シリーズ「テニス」編のすべての解説を終ります。もともとこのプログラムは，オールBASIC版を想定して作りましたので，第5章のマシン語ルーチンは，PRINT部のみに限定しました。このため期待された程は高速化が得られなかったかもしれません。これはマシン語部以外のBASIC部のためで，事実左右のラケットを同時に動かしてもほとんどスピードの落ちないことからもわかります。

今後このゲームをさらに改良するには，

① ボールとラケットのスピードを変える
② ボールの方向を45°以外にも多様化する
③ ボールとラケットとの反射角を変える
④ 障害物を設ける。
⑤ マイコン相手の1人用ゲームの追加

等が考えられます。その時，FORESIGHTの川村さんが80年の初期に「マイコン」誌に発表した「ブロック・テニス」が参考になるでしょう。この作品はPC−8001用のマシン語ゲームとして最も早く発表されたもので，PC−8001によるマシン語の使い方がまり知られていない当時の状況を考えると，その頃の一大傑作だと思います。

```
RUN
AF    BC    DE    HL    IX    IY    PC    SP
0442  D000  258A  F0A8  0000  0000  D000  CEC5
```

《第2—29図》FAC−3のアドレスを求めて

《リスト2―14》BASIC部

```
10 REM ************************
20 REM *  ｶﾞﾝｿﾘ TENNIS GAME   *
30 REM *       by K. ﾂｶｺﾞｼ    *
40 REM *        in FORESIGHT  *
50 REM *    <81.8.31-9.15>    *
60 REM ************************
99 '
100 REM ﾒｲﾝ ﾙｰﾁﾝ
105 CLEAR300,&HCFFF
106 DEF USR1=&HD000 'PRINT BALL
107 DEF USR2=&HD010 'ERASE BALL
108 DEF USR3=&HD020 'PRINT RACKET
109 DEF USR4=&HD040 'ERASE RACKET
110 WIDTH 40,25:CONSOLE 0,25,0,1:COLOR 7,0,0:PRINT CHR$(12)
115 DEFINT A-Z 'ｾｲｽｳ
120 GOSUB 1000 'ｼｮｷｾｯﾃｲ
130 GOSUB 1200 'ｼｮｷｾｯﾃｲ for RE-PLAY
140 GOSUB 1300 'ｶﾞﾒﾝ
150 GOSUB 1900 'ｻｰﾌﾞ
160 GOSUB 2000 'GAME ｦ ｵｺﾅｳ
170 IF GAME=1 THEN 150
180 GOSUB 2500 'replay ｼｮﾘ
190 IF I$="1" THEN 130
998 END
999 '
1000 REM ｼｮｷｾｯﾃｲ
1010 DIM TEN$(7,4),XMOV(4),YMOV(4)
1030 FOR I=0 TO 7
1031   FOR J=0 TO 4
1032     READ TEN$(I,J)
1033   NEXT J
1034 NEXT I
1040 READ SEN$
1050 FOR I=1 TO 4
1052   READ XMOV(I),YMOV(I)
1054 NEXT I
1090 RETURN
1099 '
1100 REM ﾃﾝ ﾉ ﾋｮｳｼﾞ
1110 COLOR 6:X=0:IF LR$="r" THEN X=34
1120 FOR I=0 TO 4
1130   LOCATE X,Y+I:PRINT TEN$(TEN,I)
1140 NEXT I
1190 RETURN
1199 '
1200 REM ｼｮｷｾｯﾃｲ for RE-PLAY
1210 GAME=0:LTEN=0:RTEN=0:LRA=14:RRA=14
1290 RETURN
1299 '
1300 REM ｶﾞﾒﾝ
1310 PRINT CHR$(12)
1320 COLOR 6:LR$="l":TEN=LTEN:GOSUB 1100 'ﾋﾀﾞﾘ PLAYER ﾄｸﾃﾝ
1325         LR$="r":TEN=RTEN:GOSUB 1100 'ﾐｷﾞ  PLAYER ﾄｸﾃﾝ
1330 COLOR 1:LOCATE 0,5:PRINT SEN$;
1335        LOCATE 0,24:PRINT SEN$;
1340 COLOR 2:LOCATE 9,1:PRINT "*******************";
1342 COLOR 7:LOCATE 9,2:PRINT "    T E N N I S    ";
1344 COLOR 2:LOCATE 9,3:PRINT "*******************";
1345 LINE( 0,6)-( 0,23)," ",3
1346 LINE(37,6)-(37,23)," ",3
1350 GOSUB 1400 'ﾋﾀﾞﾘ ﾗｹｯﾄ
1360 GOSUB 1500  'ﾐｷﾞ ﾗｹｯﾄ
1390 RETURN
1399 ;
```

105：マシン語の領域を確保する

106～109：ユーザー関数の定義

115：変数の整数化→メリット
1. 省メモリ，高速化
2. ユーザー関数の引数を整数にする

1345～1346：ラケット部のアトリビュート設定

```
1400 REM ﾋﾀﾞﾘ ﾗｹｯﾄ
1410 MM=USR3(LRA):RETURN
1499 ;
1500 REM ﾐｷﾞ ﾗｹｯﾄ
1510 XY=9472+RRA:MM=USR3(XY):RETURN
1599 ;
1600 REM BALL ｦ ｳｺﾞｶｽ
1610 XNXT=XBLL+XMOV(CRS):YNXT=YBLL+YMOV(CRS)
1620 IF XNXT<1  THEN GOSUB 2200:RETURN
1630 IF XNXT>36 THEN GOSUB 1800:RETURN
1640 IF YNXT<6  THEN GOSUB 1700:RETURN
1650 IF YNXT>23 THEN GOSUB 1700:RETURN
1660 XY=XBLL*256+YBLL:MM=USR2(XY)
1670 XBLL=XNXT:YBLL=YNXT
1680 XY=XBLL*256+YBLL:MM=USR1(XY)
1690 RETURN
1699 '
1700 REM ｺｰﾄ
1710 GOSUB 2630:ON CRS GOTO 1720,1730,1740,1750
1720 CRS=4:RETURN
1730 CRS=3:RETURN
1740 CRS=2:RETURN
1750 CRS=1:RETURN
1799 '
1800 REM ﾐｷﾞ ﾗｹｯﾄ ﾁｪｯｸ
1810 IF (YBLL-RRA)*(YBLL-RRA-2)<=0 THEN GOSUB 2610:CRS=(CRS+5)/3:RETURN
1820 LTEN=LTEN+1 'ﾋﾀﾞﾘ ﾌﾟﾚｰﾔｰ ﾎﾟｲﾝﾄ
1830 TEN=LTEN:LR$="1":GOSUB 1100
1840 GAME=1:IF LTEN=7 THEN GAME=2
1850 LOCATE XBLL,YBLL:PRINT " "
1890 RETURN
1899 ;
1900 REM ｻｰﾌﾞ
1910 COLOR 2:LOCATE 7,13:PRINT "◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆";
1920 COLOR 4:LOCATE 7,14:PRINT "     ｻ - ﾌﾞ";
1930 COLOR 7:LOCATE 7,15:PRINT "      Hit SPACE Key !";
1940 COLOR 2:LOCATE 7,16:PRINT "◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆";
1950 IF INKEY$<>" " THEN I=RND(1):GOTO  1950
1960 LINE (7,13)-(29,16)," ",7,BF
1970 GAME=0:CRS=INT(RND(1)*4)+1
1980 XBLL=18:YBLL=INT(RND(1)*12)+9:COLOR 7:LOCATE XBLL,YBLL:PRINT "●";;
1990 RETURN
2000 REM GAME ｦ ｵｺﾅｳ
2010 GOSUB 1600 'ﾎﾞｰﾙ ｦ ｳｺﾞｶｽ
2020 GOSUB 2100 'ﾗｹｯﾄ ｦ ｳｺﾞｶｽ
2030 IF GAME=0 THEN GOTO 2010
2090 RETURN
2099 '
2100 REM ﾗｹｯﾄ ｦ ｳｺﾞｶｽ
2110 K1=INP(1):K8=INP(8)
2120 IF K1=251 AND RRA>6  THEN GOSUB 2300:RRA=RRA-1:GOSUB 1500
2130 IF K1=247 AND RRA<21 THEN GOSUB 2300:RRA=RRA+1:GOSUB 1500
2140 IF K8=127 AND LRA>6  THEN GOSUB 2400:LRA=LRA-1:GOSUB 1400
2150 IF K8=191 AND LRA<21 THEN GOSUB 2400:LRA=LRA+1:GOSUB 1400
2190 RETURN
2199 '
2200 REM ﾋﾀﾞﾘ ﾗｹｯﾄ ﾁｪｯｸ
2210 IF (YBLL-LRA)*(YBLL-LRA-2)<=0 THEN GOSUB 2610:CRS=CRS*3-5:RETURN
2220 RTEN=RTEN+1 'ﾐｷﾞ ﾌﾟﾚｰﾔｰ ﾎﾟｲﾝﾄ
2230 TEN=RTEN:LR$="r":GOSUB 1100
2240 GAME=1:IF RTEN=7 THEN GAME=2
2250 LOCATE XBLL,YBLL:PRINT " "
2290 RETURN
2299 ;
```

変更：ユーザー関数に処理を委託（1400〜1510）

：ユーザー関数に変更（1660）

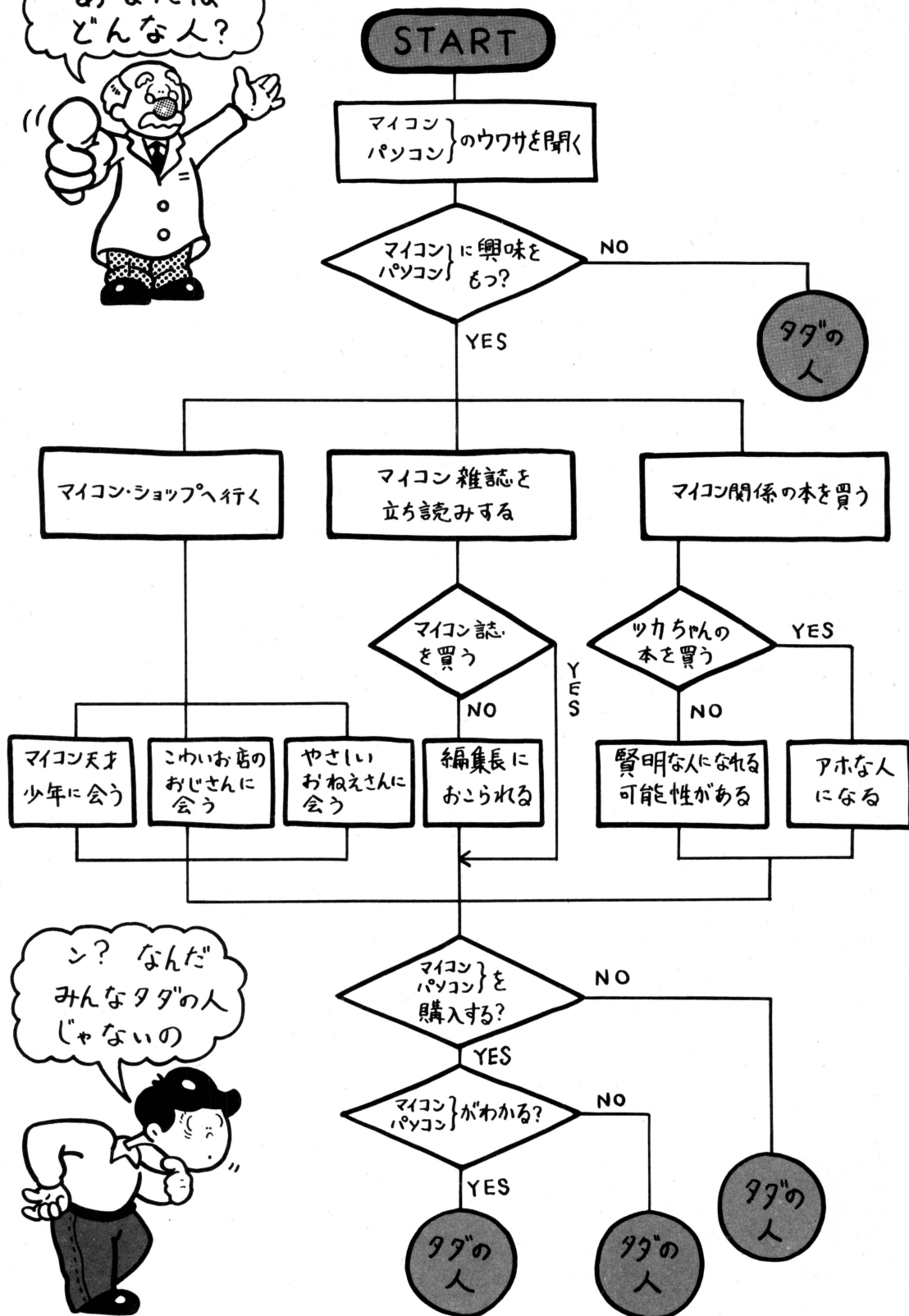
：ユーザー関数に変更（1680）

```
2300 REM ﾐｷﾞ ﾗｹｯﾄ ｹｽ
2310 XY=9472+RRA:MM=USR4(XY):RETURN
2399 ;
2400 REM ﾋﾀﾞﾘ ﾗｹｯﾄ ｹｽ
2410 MM=USR4(LRA):RETURN
2499 ;
2500 REM ｹﾞｰﾑ ｵｰﾊﾞｰ
2510 COLOR 5:LOCATE 7,13:PRINT "♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣";
2520 COLOR 2:LOCATE 7,14:PRINT "  G A M E   O V E R !  ";
2530 COLOR 7:LOCATE 7,15:PRINT "     1･･･r e p l a y   ";
2540         LOCATE 7,16:PRINT "     2･･･s t o p       ";
2550 COLOR 5:LOCATE 7,17:PRINT "♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣";
2560 I$=INKEY$:IF I$<>"1" AND I$<>"2" THEN 2560
2570 LINE (7,13)-(29,17)," ",7,BF
2590 RETURN
2599 '
2600 REM ｵﾄ
2610 BEEP 1:FOR TT=0 TO 40:BEEP 0:NEXT
2620          FOR TT=0 TO 20:NEXT
2630 BEEP 1:FOR TT=0 TO 50:BEEP 0:NEXT
2690 RETURN
2699 '
9000 DATA "◢■◣"
9010 DATA "■ ■"
9020 DATA "■ ■"
9030 DATA "■ ■"
9040 DATA "◥■◤"
9050 DATA " ◢ "
9060 DATA "  ■ "
9070 DATA "  ■ "
9080 DATA "  ■ "
9090 DATA " ◢◣"
9100 DATA "◢■◣"
9110 DATA "■ ■"
9120 DATA "  /"
9130 DATA " / "
9140 DATA "◢■■"
9150 DATA "◢■◣"
9160 DATA "■ ◤"
9170 DATA "  ■ "
9180 DATA "■ ◣"
9190 DATA "◥■◤"
9200 DATA "◢  "
9210 DATA "■ ■ "
9220 DATA "■ ■ "
9230 DATA "◥■◣"
9240 DATA "  ■ "
9250 DATA "■■◣"
9260 DATA "■  "
9270 DATA "◥■◣"
9280 DATA "   ■"
9290 DATA "◥■◤"
9300 DATA "◢■◣"
9310 DATA "■   "
9320 DATA "■■◣"
9330 DATA "■ ■"
9340 DATA "◥■◤"
9350 DATA "■■■"
9360 DATA "■ ■"
9370 DATA "   ■"
9380 DATA "  /"
9390 DATA " / "
9400 DATA "■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■"
9410 DATA 1,-1,-1,-1,-1,1,1,1:' BALL ﾉ ﾍﾝｲ
```

変更：ユーザー関数に処理を委託

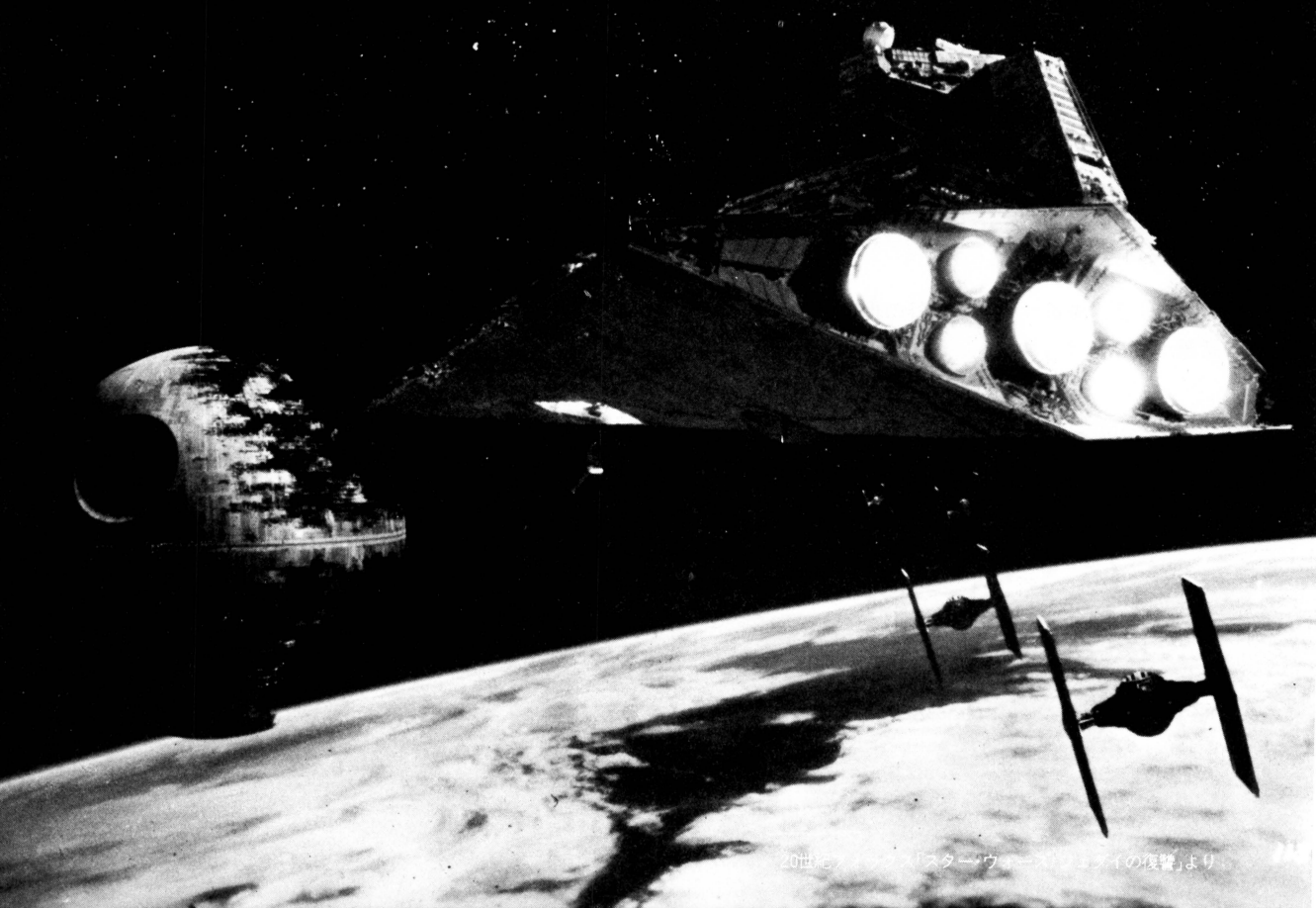
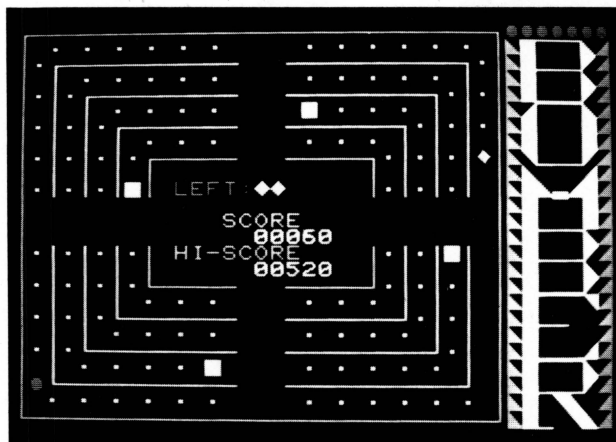
《リスト2—15》マシン語部

```
D000 2AA8F0     LD   HL,(0F0A8H)   :HL←引数(ユーザー関数よりボールの位置を得る)
                                                                    ボール出力
D003 24         INC  H             ⎫
D004 2C         INC  L             ⎬ アドレス変換
D005 CDF303     CALL 03F3H         ⎭
D008 36EC       LD   (HL),0ECH     :その位置にボールを書込む
D00A C9         RET
D00B 00         NOP                ⎫ ダミー
D00C 00         NOP                ⎪
D00D 00         NOP                ⎪
D00E 00         NOP                ⎪
D00F 00         NOP                ⎭
D010 2AA8F0     LD   HL,(0F0A8H)
                                                                    ボール消去
D013 24         INC  H
D014 2C         INC  L
D015 CDF303     CALL 03F3H
D018 3620       LD   (HL),20H      :ボールの消去
D01A C9         RET
D01B 00         NOP                ⎫ ダミー
D01C 00         NOP                ⎪
D01D 00         NOP                ⎪
D01E 00         NOP                ⎪
D01F 00         NOP                ⎭
D020 2AA8F0     LD   HL,(0F0A8H)   ⎫
D023 24         INC  H             ⎬ HL←LOCATE座標
D024 2C         INC  L             ⎭     ただし,(1,1)オリジンとする
D025 0603       LD   B,03H         :カウンタ, 3キャラクタ分
D027 C5         PUSH BC
D028 E5         PUSH HL
D029 CDF303     CALL 03F3H         :アドレス変換
D02C 3687       LD   (HL),87H      :ラケットの書込み"■"1キャラクタ分
D02E E1         POP  HL
D02F 2C         INC  L             :一つ下の行へ
D030 C1         POP  BC
D031 10F4       DJNZ 0D027H
D033 C9         RET
D034 00         NOP                ⎫
D035 00         NOP                ⎪
D036 00         NOP                ⎪ ダミー
D037 00         NOP                ⎪
D038 00         NOP                ⎪
D039 00         NOP                ⎪
D03A 00         NOP                ⎪
D03B 00         NOP                ⎪
D03C 00         NOP                ⎪
D03D 00         NOP                ⎪
D03E 00         NOP                ⎪
D03F 00         NOP                ⎭
D040 2AA8F0     LD   HL,(0F0A8H)
                                                                    ラケット消去
D043 24         INC  H
D044 2C         INC  L
D045 0603       LD   B,03H
D047 C5         PUSH BC
D048 E5         PUSH HL
D049 CDF303     CALL 03F3H
D04C 3620       LD   (HL),20H      :消去(1キャラクタ分)
D04E E1         POP  HL
D04F 2C         INC  L
D050 C1         POP  BC
D051 10F4       DJNZ 0D047H
D053 C9         RET
```

あなたは どんな人？
START
マイコン パソコン のウワサを聞く
マイコン パソコン に興味をもつ？
NO
タダの人
YES
マイコン・ショップへ行く
マイコン雑誌を立ち読みする
マイコン関係の本を買う
マイコン誌を買う
YES
NO
ツカちゃんの本を買う
YES
NO
マイコン天才少年に会う
こわいお店のおじさんに会う
やさしいおねえさんに会う
編集長におこられる
賢明な人になれる可能性がある
アホな人になる
マイコン パソコン を購入する？
NO
YES
マイコン パソコン がわかる？
NO
YES
タダの人
タダの人
タダの人
ン？ なんだ みんなタダの人 じゃないの

第3ブロック

リアルタイムゲーム「BOMBER」に挑戦

20世紀フォックス「スター・ウォーズ ジェダイの復讐」より

リアルタイムゲーム「BOMBER」に挑戦

オリエンテーション

はじめに

“雪がとけて　川になって
流れて行きます
つくしの子が　恥ずかしげに
顔を出します
もうすぐ春ですね”

キャンディーズが解散してはや4年，かくてここに「**GAMINGへの招待——第3ブロック**」が始まろうとしています。

マルチプログラミング

突然ですが，最近マイコン以外の世界でもマイコンの話題が取り上げられ，

果たしてマイコンで何が出来るか？
マイコンでスモール・ビジネスが可能か？

なんて論じられているのを良く見かけますね。そこで

大型機とマイコン

を比べてみることにしましょうか？

第3－1図を御覧ください。大型機とマイコンの決定的に違う点は，大型機における

マルチプログラミング(multi programming)

にあります。マルチプログラミングというのは，機械は1台なのに同時に何本ものプログラムを走らせることで，CPUの処理時間を細かく区切って対処しています。このため対他人との関係が生じ，**自由にシステムを使うのが難しく**なっています。たとえば沢山のプログラムを同時に走らせれば，やはり個々のプログラムの処理速度は落ちますし，磁気テープ装置の奪い合いということも起きますし，急ぎの仕事なのにプリン

大　型　機	←――→	パーソナル・コンピュータ（マイコン）
超高速，信頼性大 カタカナどまり	プ　リ　ン　タ	英大小文字，カタカナ，ひらがな グラフィックス，なんと漢字まで使用可
メイン・コンソール(or端末のディスプレイ) 英大文字のみ	C　R　T	高解像度カラー・グラフィックス
あり そのため，対他人との相関関係が現われる	マルチプロミラミング	なし 自由自在に使える
⋮	⋮	⋮
プリンタを想定した静的なもの マイコンの世界では，過去の遺物	GAME	カラー・グラフィックスによるダイナミックな動きのあるリアル・タイム・ゲーム

(注)　この表における大型機の周辺装置は，その性能及び用途から一般にこのように使われることが多いことを示したにすぎない。したがって大型機の周辺装置の性能が悪いと思われては困る。値段が何ケタも異なるのだから，その性能，信頼性の高いのは当り前。たとえば普通のマイコン用ドッド・インパクト・プリンタで1時間もかかるようなアセンブル・リストでも，大型機の高速プリンタならさく岩機でもかけているような騒音をたてながらあっという間に終ってしまう。

《第3－1図》大型機 対 パーソナルコンピュータ

タが塞がっていて待たされるということもあります。

今でこそ

仮想記憶 (virtual memory)

が普及してメモリ制限等無くなりましたが，以前は一度にコンパイルは3本までとか，あまりメモリを使うと他人におこられるということがありました。

その点，マイコンは気楽なものです。あなたのマシンです。自由にGAMEでも作って遊びましょう。ガハハハハハ。

さあ，そこで第3ブロックもGAME作りを楽しむことにします。何を取り上げるかいろいろと迷いました。ハテ，サテ，コリャコリャ。そして結局第3ブロックも

リアル・タイムGAME

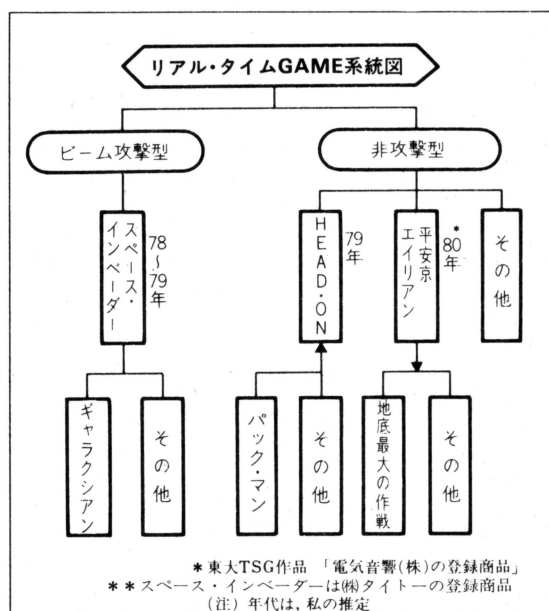
に挑戦することにしました。というのは**第3－1図**を見てもわかるように，**周辺装置**ではマイコンも結構頑張っています。その中でも

カラーグラフィックスCRT

なかなか張り切っていますね。そのカラーグラフィックスを最高に活かすGAMEは，何と言ってもリアル・タイムGAMEですね（なんてカッコつけてますが，本当はクラブの中で，リアル・タイムGAMEが一番人気があるのです)。

さあ，そこでリアル・タイムGAMEの中でどのGAMEを取り上げるかですが，今回は一世を風靡したあの **"スペース・インベーダー"** とはひと味異なる

BOMBER

というゲームです。

《第3－2図》リアル・タイムGAMEの起源・分類

GAMING基本3原則

さて，第3ブロックもGAME作りにあたって次の基本3原則でのぞみたいと思います。

① ＢＡＳＩＣの基本コマンド卒業程度の人をはじめとして，**できるだけ幅広い層を対象**とする。まだそこまできていない人は，とりあえず保存しておいて後日役立ててください。

② 使用言語はできるだけ**標準ＢＡＳＩＣ**とする。やむえずＰＣ－8001の方言を使うときは，その意味に言及する。

③ **プログラムの作成は，あなた自身がおこなう。**説明の都合上プログラム・リストはのせますが，あくまでも参考にするにとどめ，現在のあなたが理解出きる命令だけを使ってあなた自身の手により自分のプログラムを作成してください。

プログラム作りへのお誘い

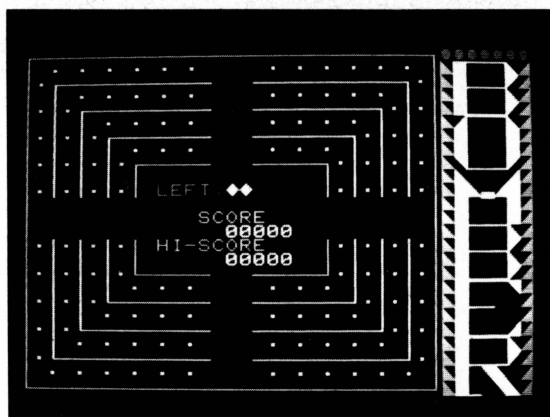
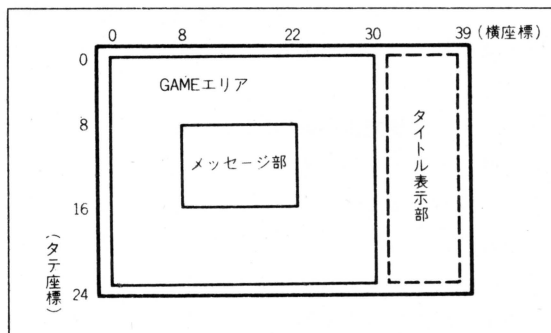
さあ，以上を考慮した上で**写真１**を御覧ください。これが今回作っていこうとする'ＢＯＭＢＥＲ'の画面です。ＰＣ－8001で作ったもので，ＰＣのカラー機能を生かすべく配色には気をくばったつもりです。さらに**リスト３－７**がそのプログラムです。**リスト３－７**のプログラムをキー・インし，**ＲＵＮ**させれば**写真１**の画面が得られます。

さて，あなたは**写真１**およびプログラム・リストを見てどうお感じになったでしょうか？　まさにＧＡＭＥの画面だし，ＧＡＭＥのプログラムですね。もしあなたが写真を見て

「ヘボイ画面だ，オレならこう作る」

と思われたなら結構，ただちに**あなたのマシンであなたのプログラム**を作ってみてください。なにせ今回の目標は**写真１**に準じた

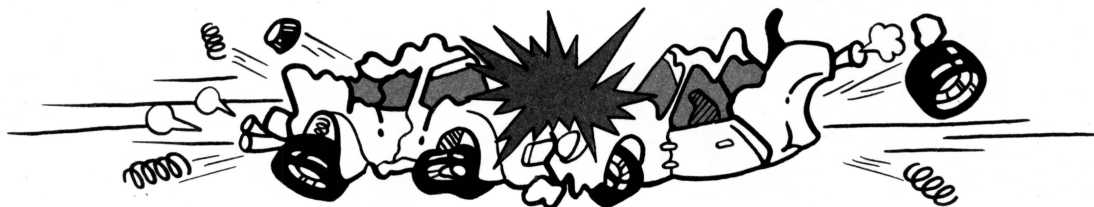
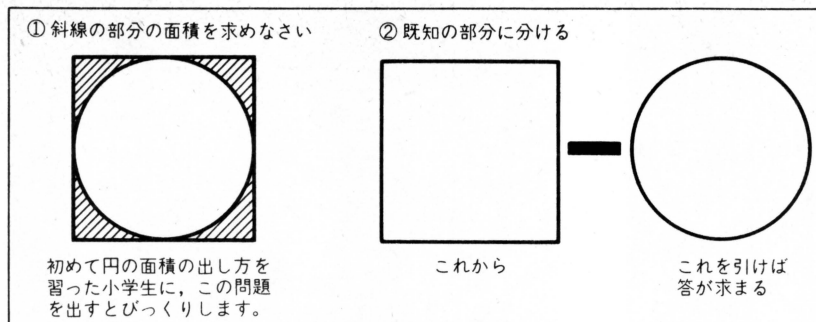
'ＢＯＭＢＥＲ' のＧＡＭＥ画面を作る

ことですから。なお参考までに**写真１の画面レイアウト**を，**第３－３図**に示しておきます。

《写真１》BOMBER完成画面

《第３－３図》BOMBER画面レイアウト

さて，不幸にして**写真１**を見て難しそうだと思われた人は，以下の小文が参考になるかもしれません。その後なんとか自分の知っている命令の範囲で，**写真１**のような画面を作ってみてください。きっとプログラム作りが楽しくなることでしょう。

ＰＣ－8001以外のマシンをお持ちの方に，１点だけ御注意申し上げておきます。**リスト３－７**の220行とか1010行とかに，見慣れない命令があるかもしれません。しかしその部分はＰＣ特有の命令で，画面モードとか色とか，ＰＣのハードウエアに関する命令です。したがって無視していただいて影響はありません。

合成写真の妙技

第３－４図を御覧ください。おなじみ小学校での面

《第3－4図》算数の問題（面積を求める）

積を求める問題です。**'円の面積を求める公式'** を習ったばかりの小学生に①の問題を見せると，天地がひっくり返ります。しかし②のように，

〈正方形の面積〉－〈円の面積〉

で求まることを教えれば，なる程と納得します。

すなわち

困難を分解し，既知のものへ帰着！

させるわけです。

写真1の画面にしても同じです。マイコンを始めて間もない方には，全体を一度に見てしまうため，一見難しく見えるかもしれません。しかし，**写真1**の画面はもともと

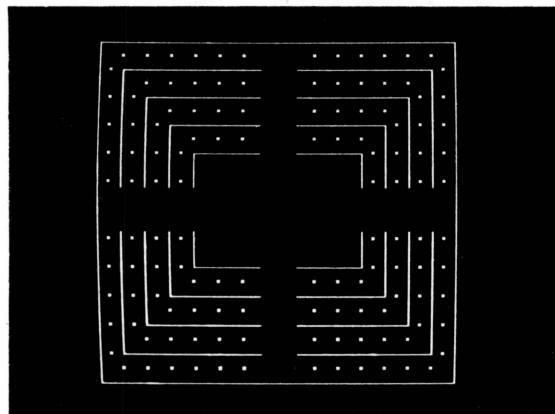
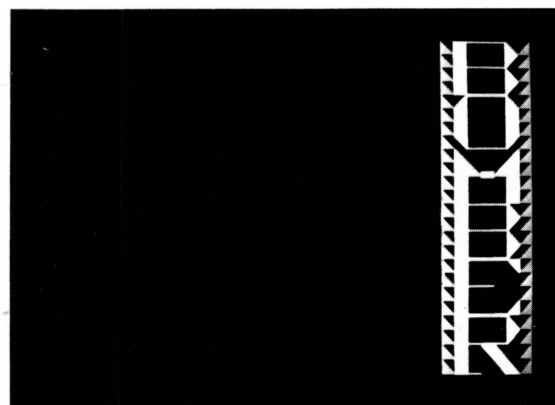
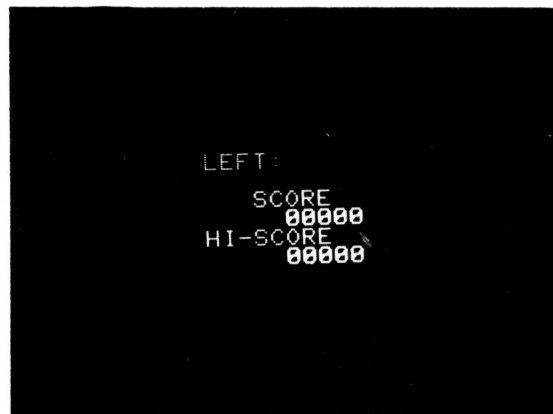
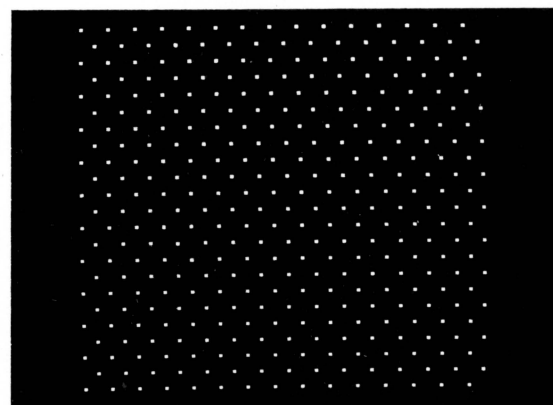



写真2：GAME エリア

写真3：タイトル

写真4：各種メッセージ

を合成して作ったものなのです。この中でもっとも面倒な**写真2**は，さらに分解され

写真5：ドットの表示

写真6：ワクの表示

写真7：**写真6**より不要部を消去したもの

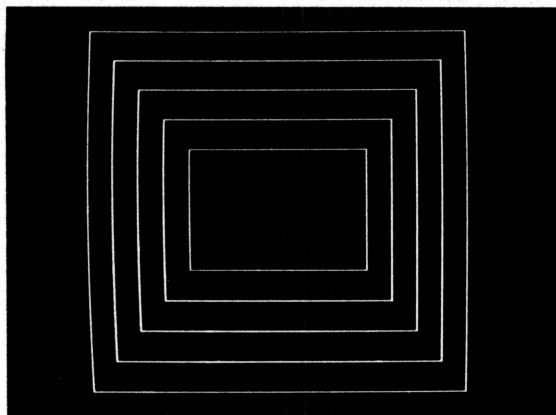
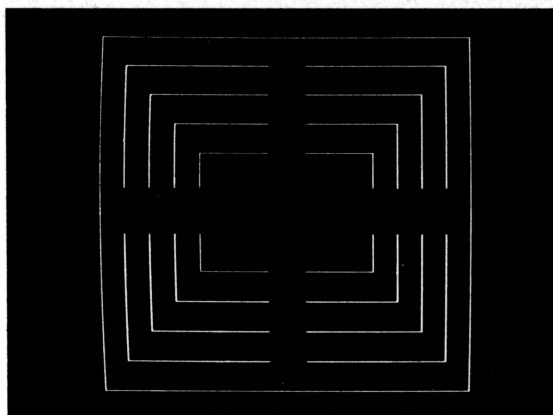
により構成されているのです。なお**写真7**を補足説明するため**写真8**を入れておきます。これは，画面いっぱいに♥を描いたあと**写真7**と同じ'消去ルーチン'を用いて不要部分を消去したものです。

さあ，どうですか？　初め複雑に見えた**写真1**も，もともとは**簡単な部分を合成**して作ったものなのです。これらの部分なら簡単にプログラム出来そうに思いませんか？

《写真2》GAME部

《写真4》メッセージ部

《写真3》タイトル部

《写真5》ドットの表示

《写真6》ワクの表示

《写真7》消　去

オワリ名古屋はPRINT文

一例を示しましょう。

PRINT　"―"

の命令はわかりますね。これを実行すると第3―5図①のようになります。これを

FOR～NEXT

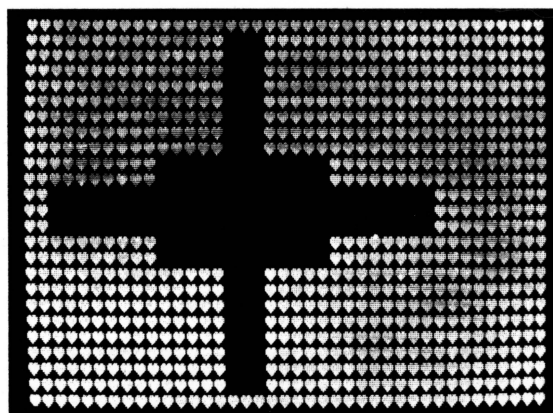
を使って沢山書けば②のようになります。それを2本組み合わせたのが③ですね。同様に

PRINT　"|"

から⑤ができます。③と⑤を合わせれば，⑥ですね。これに

LOCATE

で

《第3―5図》長方形を作る

《写真8》写真7の補足

「，┐，└，」

をつけてやれば長方形の完成です（なおLOCATEはPC―8001におけるPRINT位置を指定する命令です)。さらにこの長方形を小さいのから大きいのまで組み合わせれば，**写真6**の完成となるわけです。

以上のようにある複雑な図形をプログラムしようとしたら，より簡単な図形に分解してみてください。それでも難しかったら，さらにその図形を分解してみてください。こうして分解を繰り返していけば，いつかはあなたのプログラム可能な図形に到達することでしょう。なにしろ最終的な段階までに分解すれば，

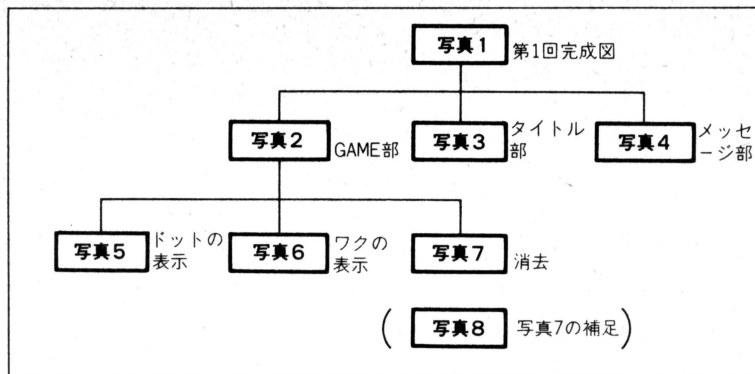
PRINT　"チョメチョメ"

になってしまうのですから。

まとめると

それでは，そろそろ第1章のまとめに入りましょう。

各写真の構成を第3―6図にまとめておきます。また各写真とリストの関係をまとめておくと，

リスト3―1：写真3
リスト3―2：写真4

《第3－6図》写真構成図

リスト3－1　タイトルの表示

```
220 WIDTH 40,25:CONSOLE 0,25,0,1:COLOR 5,32,0:PRINT CHR$(12);
1070 FOR Y=1 TO 24                                  'PRINT TITLE
1072   READ I$:COLOR 2:LOCATE 31,Y:PRINT "◣";:COLOR 6:PRINT I$;  ⎫1行分の処理
1073   COLOR 2:PRINT "◢";                                          ⎭
1074 NEXT
1090 GOTO 1090 ←無限ループ
```

リスト3－2　メッセージ表示部

```
220 WIDTH 40,25:CONSOLE 0,25,0,1:COLOR 5,32,0:PRINT CHR$(12);
1080 COLOR 2:LOCATE 10,10:PRINT "LEFT:    ";:GOSUB 1200 'PRINT MESSAGE
1082 COLOR 4:LOCATE 13,12:PRINT    "SCORE";:GOSUB 1300
1084 COLOR 4:LOCATE 10,14:PRINT "HI-SCORE";:GOSUB 1400
1090 GOTO 1090
1099 '
1200 '—— PRINT LEFT ——
1210 IF LEFT=0 THEN RETURN                     'LEFT=0 THEN ヒョウジ セズ
1220 FOR X=15 TO 14+LEFT
1230   COLOR 7:LOCATE X,10:PRINT "●";
1240 NEXT:RETURN
1299 '
1300 '—— PRINT SCORE ——
1310 COLOR 7:LOCATE 15,13:PRINT RIGHT$("0000"+HEX$(SCR),5):RETURN
1399 '
1400 '—— PRINT HI-SCORE ——
1410 COLOR 7:LOCATE 15,15:PRINT RIGHT$("0000"+HEX$(HISCR),5):RETURN
```

リスト3－3　ドットの表示

```
220 WIDTH 40,25:CONSOLE 0,25,0,1:COLOR 5,32,0:PRINT CHR$(12);
1020 FOR Y=1 TO 23
1022   X=(Y MOD 2)+1
1024   FOR X=X TO X+28 STEP 2
1026     LOCATE X,Y:PRINT "・";
1028 NEXT X,Y
1090 GOTO 1090
```

リスト3－4　ワクの表示

```
220 WIDTH 40,25:CONSOLE 0,25,0,1:COLOR 5,32,0:PRINT CHR$(12);
1030 FOR I=0 TO 8 STEP 2                          'PRINT RECTANGLE
1031   IF (I MOD 8)=0 THEN COLOR 1 ELSE COLOR 5
1032   LOCATE I,I:PRINT "┌";
1033   Y1=I:Y2=24-I
1034   FOR X=I+1 TO 29-I
1036     LOCATE X,Y1:PRINT "─";:LOCATE X,Y2:PRINT "─";
1038   NEXT
1039   LOCATE X,Y1:PRINT "┐";:LOCATE X,Y2:PRINT "┘";
1040   X1=I:X2=30-I
1042   FOR Y=I+1 TO 23-I
1044     LOCATE X1,Y:PRINT "│";:LOCATE X2,Y:PRINT "│";
1046   NEXT
1048   LOCATE X1,Y:PRINT "└";
1050 NEXT
1090 GOTO 1090
```

リスト3－5　消去

```
1060 X1=9 :X2=21:Y1=9 :Y2=15:GOSUB 1100         'CROSS OUT
1062 X1=14:X2=16:Y1=1 :Y2=23:GOSUB 1100
1064 X1=1 :X2=29:Y1=11:Y2=13:GOSUB 1100
1090 GOTO 1090
1100 '--- CLEAR RECTANGLE ----------
1101 ' ---- PARA IN:X1,X2,Y1,Y2 --
1110 FOR Y=Y1 TO Y2
1120   FOR X=X1 TO X2
1130     LOCATE X,Y:PRINT " ";
1140 NEXT X,Y:RETURN
1199 '
```

リスト3－6　♥を消去する

```
20 WIDTH 40,25:CONSOLE 0,25,0,1:COLOR 7,&HE9:PRINT CHR$(12);
30 FOR I=0 TO 2000:NEXT
```

《リスト3－7》BOMBERプログラムリスト（第1章）

```
10 '♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
12 '♥  BOMBER for 'INVITATION FOR GAMING'2-1 ♥
14 '♥               << 82.2.15-X.XX >>        ♥
16 '♥                       by K.TUKAGOSHI ♥
18 '♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
49 '
50 'HISCR              :ﾊｲ ｽｺｱ
52 'LEFT               :COUNTER OF LEFT CAR
54 'SCR                :ｽｺｱ
99 '
100 '▬▬ MAIN ROUTINE ▬▬
110 '--- COLD START ---
120 HISCR=0
199 '
200 '--- HOT  START ---
210 SCR=0:LEFT=3
220 WIDTH 40,25:CONSOLE 0,25,0,1:COLOR 5,32,0:PRINT CHR$(12);
230 GOSUB 1000                                 'ｶﾞﾒﾝ
900 GOTO 900                                   'MAIN END
999 '
1000 '▬▬ SUB ROUTINE ▬▬
1001 '--- ｶﾞﾒﾝ ---
1010 COLOR 5:PRINT CHR$(12);                    'PRINT
1020 FOR Y=1 TO 23
1022   X=(Y MOD 2)+1
1024   FOR X=X TO X+28 STEP 2
1026     LOCATE X,Y:PRINT "・";
1028 NEXT X,Y
1030 FOR I=0 TO 8 STEP 2                        'PRINT RECTANGLE
1031   IF (I MOD 8)=0 THEN COLOR 1 ELSE COLOR 5
1032   LOCATE I,I:PRINT "┌";
1033   Y1=I:Y2=24-I
1034   FOR X=I+1 TO 29-I
1036     LOCATE X,Y1:PRINT "─";:LOCATE X,Y2:PRINT "─";
1038   NEXT
1039   LOCATE X,Y1:PRINT "┐";:LOCATE X,Y2:PRINT "┘";
1040   X1=I:X2=30-I
1042   FOR Y=I+1 TO 23-I
1044     LOCATE X1,Y:PRINT "│";:LOCATE X2,Y:PRINT "│";
1046   NEXT
1048   LOCATE X1,Y:PRINT "└";
1050 NEXT
1060 X1=9 :X2=21:Y1=9 :Y2=15:GOSUB 1100         'CROSS OUT
1062 X1=14:X2=16:Y1=1 :Y2=23:GOSUB 1100
1064 X1=1 :X2=29:Y1=11:Y2=13:GOSUB 1100
1070 FOR Y=1 TO 24
1072   READ I$:COLOR 2:LOCATE 31,Y:PRINT "◣";:COLOR 6:PRINT I$;
1073   COLOR 2:PRINT "◢";
1074 NEXT
1080 COLOR 2:LOCATE 10,10:PRINT "LEFT:    ";:GOSUB 1200 'PRINT MESSAGE
1082 COLOR 4:LOCATE 13,12:PRINT      "SCORE";:GOSUB 1300
1084 COLOR 4:LOCATE 10,14:PRINT "HI-SCORE";:GOSUB 1400
1090 RETURN
1099 '
1100 '--- CLEAR RECTANGLE ----------
```

```
1101 ' ──── PARA IN:X1,X2,Y1,Y2 ──
1110 FOR Y=Y1 TO Y2
1120   FOR X=X1 TO X2
1130     LOCATE X,Y:PRINT " ";
1140 NEXT X,Y:RETURN
1199 '
1200 '─── PRINT LEFT ───
1210 IF LEFT=0 THEN RETURN                              'LEFT=0 THEN ﾋｮｳｼﾞ ｾｽﾞ
1220 FOR X=15 TO 14+LEFT
1230   COLOR 7:LOCATE X,10:PRINT "●";
1240 NEXT:RETURN
1299 '
1300 '─── PRINT SCORE ───
1310 COLOR 7:LOCATE 15,13:PRINT RIGHT$("0000"+HEX$(SCR),5):RETURN
1399 '
1400 '─── PRINT HI-SCORE ───
1410 COLOR 7:LOCATE 15,15:PRINT RIGHT$("0000"+HEX$(HISCR),5):RETURN
1499 '
9000 '▬▬▬ DATA AREA ▬▬▬
9010 '─── TITLE DATA ───
9020  DATA "█▔▔◣"
9022  DATA "█▁▁◤"
9024  DATA "█  ◣"
9026  DATA "█▁▁◤"
9028  DATA "◢▔▔◣"
9030  DATA "█  █"
9032  DATA "█  █"
9034  DATA "◥▁▁◤"
9036  DATA "◣  ◢"
9038  DATA "█◣▁◢█"
9040  DATA "█ ▔ █"
9042  DATA "█  █"
9044  DATA "█▔▔◣"
9046  DATA "█▁▁◤"
9048  DATA "█  ◣"
9050  DATA "█▁▁◤"
9052  DATA "█▔▔◥"
9054  DATA "█▁  "
9056  DATA "█   "
9058  DATA "█▁▁◢"
9060  DATA "█▔▔◣"
9062  DATA "█▁▁◤"
9064  DATA "█ ◥ "
9066  DATA "█▁ ◥"
```

リスト３－３：写真５

リスト３－４：写真６

リスト３－５：消去部のルーチン

リスト３－６：このあとにリスト３－５を続けると写真８が得られる

なお今回のプログラムは，できる限り標準ＢＡＳＩＣを意識したためＰＣのオーナーには少し歯痒かったかもしれません。たとえば**ＬＩＮＥ文**を使えば長方形なんて一発で書けますね。

おわりに

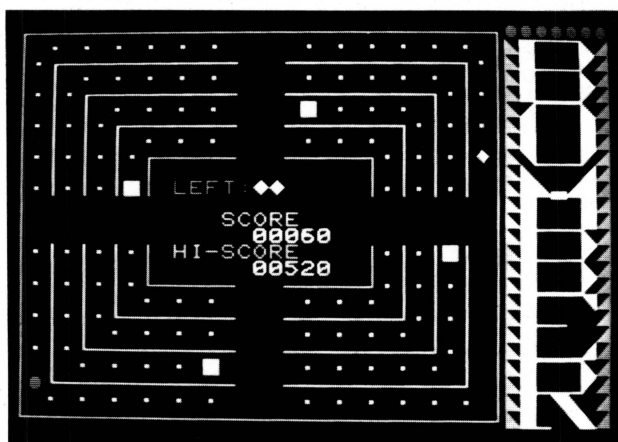
第１章いかがだったでしょうか？　何かしら吸収できるものがあったなら幸いに思います。たとえば初めてあなたが，長方形のＰＲＩＮＴに成功したとします。するとそれは**あなた自身がプログラム**したことになります。なぜなら第１章で，長方形のプログラミングの**考え方は説明**しましたが，**プログラムそのものの説明はしてなかった**はずです。したがって長方形のプログラムができたなら，それはあなた自身が組んだことになります。これぞ

塚越式寺小屋

だ～！

ところで’ＢＯＭＢＥＲ’，’ＢＯＭＢＥＲ’と騒ぎたてましたが，’ＢＯＭＢＥＲ’って何ですか？どうやって遊ぶのですか？　第２章以降をお楽しみに。

“もうすぐ　は～～るですね”

リアルタイムゲーム「BOMBER」に挑戦
GAME仕様の分析

If the rain comes,
They run and hide their head,
They might as well be dead,
If the rain comes.

（美糸留守）

はじめに

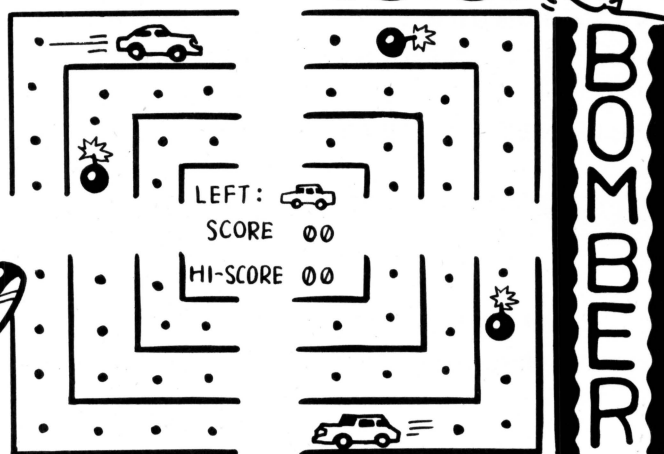
水無月――全国的に六月がおとづれ，梅雨の季節となります。いかがお過ごしですか？

このうっとおしい季節をむかえ，かくて〝**GAMING への招待**〟――〝**BOMBER**〟第2章のはじまりです。第1章の画面制作，うまくいきましたか？

うまくいったと勝手に解釈し，第2章はその続きを作っていくことにします。

ところで，6月，私の誕生月でもあります。

そこで――。

Happy birthday to me,
Happy birthday to me!

GAMEの遊び方

さて，TV画面が出来上がっていますので，これからGAME作りの本体へと進んで行くわけです。しかし，まだ〝BOMBER〞というGAMEの遊び方は紹介していませんでした。そこで，このGAMEの遊び方を簡単に説明しておきます。

まず，**第3－7図**が第1章で作ったTV画面です。

GAMEをスタートさせると，**第3－8図**のようになります。ここで新しいキャラクタが，二つ登場してきました。

●(白)：マイカー

♥(赤)：レッド・カー

もしあなたのマシンでグラフィックが使えるなら，もっと車らしいグラフィック・パターンに変えてみてください。ここでは上記のようにあなたの車の敵の車を設定してあります。

さて，そのままにしておくと両者の車は，

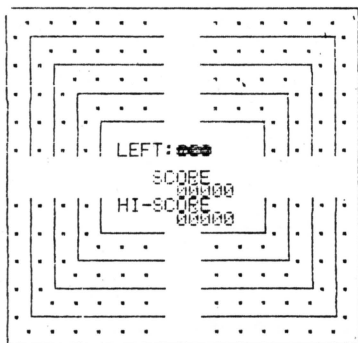
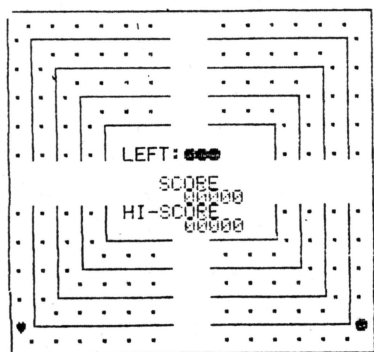
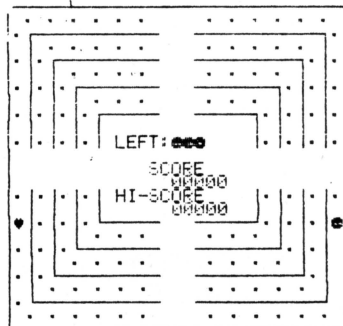
●（味方）：左まわり

♥（敵）　：右まわり

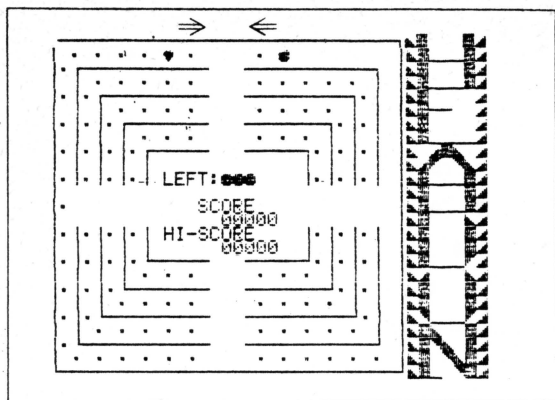
のように動き始めます（第3－9図）。

ところで**第3－9図**を良く御覧になってください。あなたの車の動いた跡には，**ドット・**が消えていますね。そうです，この

GAMEの目的は

味方の車をじょうずに操作して，できるだけ沢山のドット・を消すことにあります。

《第3－7図》第1章の最終版

《第3－8図》GAMEをスタートさせる

《第3－9図》車が動き出す

《第３―10図》放っておくと……

《第３―11図》正面衝突！

ところでただドット・を消すだけなら簡単です。しかし，そこはGAMEです。そうあまくはありません。そのまま車を放っておくと，両者ともコースの外側を通り，やがて近づいて行きます（第３―10図）。そして，

衝　突　！

してしまいます（第３―11図）。そうするとあなたの車は，**クラッシュしてしまうのです！**

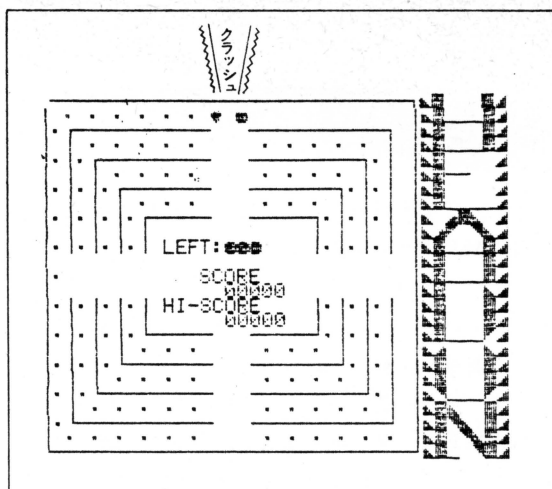
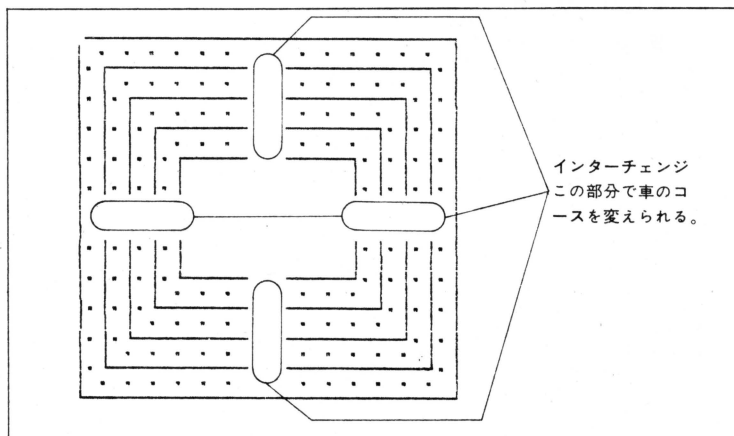
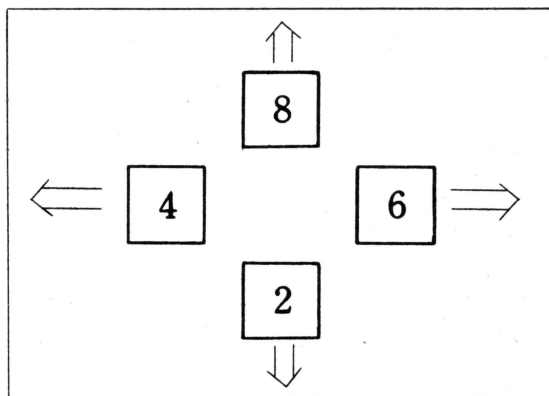
そこで，このクラッシュを避けるため，第３―12図のように各**インターチェンジ**を利用して，コースを変えます。ところが敵もさるものひっかくもので，ちゃんとこちらの動きをキャッチしながらコースを変えてきます。ですから，いかにコースを変えるかがあなたの腕の見せどころです。

《第３―12図》インターチェンジ

以上がこのGAMEの遊び方の基本です。他にもGAMEを面白くするためのいろいろなバリエーションがあります。第3ブロックで，それらのバリエーションを取り入れるかは，まだ未定です。余裕があれば取り入れたいと思います。マイカーのキー操作は，私のマシンでは第３―13図のようにしました。また，ドット・一つを何点にするか等の得点類については，GAMEができあがっていく段階でおいおい決めたいと思います。

《第３―13図》キー・ファンクション

悲喜こもごも

ビデオ・ゲームが，いま二億アメリカ人をとりこにしている，おなじみスペース・インベーダーのあと，続々とヒット作が登場した。……人気抜群の台をのぞくと，迷路上のカラー画面に，頭と足だけの人形がひょこひょこ。いまはやりのパックマンだ。白い点々を食べて点数を増やす。ぎょろ目のモンスターが人形をとって食おうとする。

――“読売新聞”（57年４月３日）

マイコンで動きのあるGAMEを作ろうと思ってシ

コシコ頑張っているんだけど，難しいね。とくに絵の動かし方が良くわからない。やっと

FOR～NEXT

を使ってタマを打ち上げるのに成功した。そうしたら今度は，ビーム砲が動かない。**二つのものを同時に動かす**には，どうしたらいいんでしょうね？　私？　今〝スペース・インベーダー〟を作っています。

（Y氏。大型コンピュータ歴7年。マイコン購入後，2か月。）

ところで皆さん！日本のゲーム・センター，高いと思いませんか？　たとえばパリでは，昨秋値段が倍化され，ゲーム・センター熱が下がったそうです。しかしそれでもまだ80円（2フラン）です（**第3―14図**）。

ニューヨーク	25セント	（60円）
ロンドン	20ペンス	（100円）
パ　リ	2フラン	（80円）
ボ　ン	1マルク	（100円）
ニューデリー	1ルピー	（26円）
香　港	1香港ドル	（40円）
マニラ(禁止)	1ペ　ソ	（30円）

〝読売新聞〟（57年4月3日）より

《3―14図》世界の主な都市のゲーム代

位置変数の導入

さあ，いよいよマイカーを登場させ，GAMEエリア上を動かしますよ。

マイカーをドット・上に動かし，
動いたあとドット・を消す――――

どうやったらいいのでしょうね。?

第1章で，複雑な画面を作るのに

困難を分割

しました。第2章でも同じです。いきなりできあがったGAMEを見ると，いかにも難しく見えます。でも作る側から見るとたいしたことはありません。**簡単なことを積み上げて行っただけ**ですから，

そこで困難を分割してみます。

複雑なGAMEエリアを単純化し，**第3―15図**のように一直線のドットを考えます。このドット上でマイカーを動かすのです。

一般に**図形P**を動かしたいと思ったら

その図形の位置を
変数X：ヨコ座標
変数Y：タテ座標

で表わすとうまくいきます。

そこでマイカーの位置を表わすのに

XMYCAR：ヨコ座標
YMYCAR：タテ座標

の2変数を導入することとしましょう。そして，それらの変数に初期値を与えてやります。たとえば，

XMYCAR＝10
YMYCAR＝24

のように。これに行番号を与えてやれば，立派なBASICの代入文になります。

```
10  XMYCAR=10
20  YMYCAR=24
```

この変数を使って**第3―15図**のドットを描くには，**リスト3―8**でできます。また，**リスト3―9**のPRINT文でマイカーを表示できます。それを表示したのが**第3―16図**です。

マイカーを動かす

次にこのマイカーを動かしてみましょう。下から上までドットを消しながら動かします。方法は簡単で次の通りです。

① マイカーを消す

```
LOCATE XMYCAR, YMYCAR
PRINT " ";
```

で消えますね。

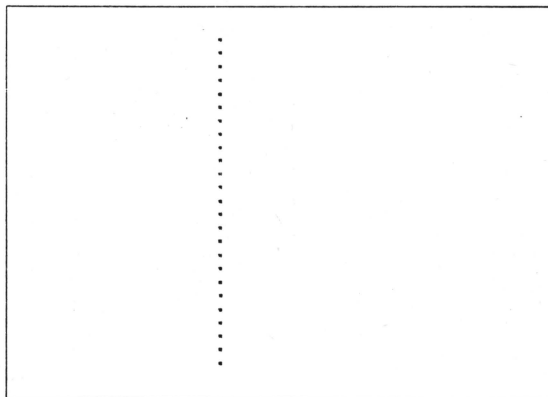
② マイカーの座標を一つ上にする

マイカーのタテ座標YMYCARを一つ上（すなわち一つ小さくする）にすれば良いから，

```
YMYCAR=YMYCAR-1
```

を実行します。

③ そこにマイカーを書きます。

```
LOCATE XMYCAR, YMYCAR
PRINT "●";
```

《第3―15図》直線だけで考える

《リスト3－8》ドット表示

```
110 XMYCAR=10:YMYCAR=24                    'INITIAL
120 FOR Y=0 TO YMYCAR                      'LINE-0 TO LINE-24
130   LOCATE XMYCAR,Y:PRINT "・";
140 NEXT
900 GOTO 900                               'FOR LOOP
```

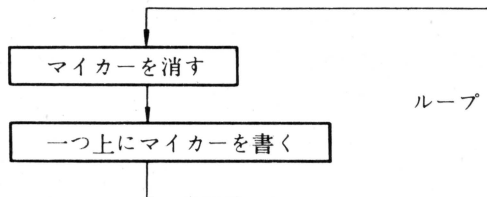
《リスト3－9》マイカー表示

```
150 LOCATE XMYCAR,YMCAR:PRINT "●";
```

これでマイカーが一つ上にあがりました。以下

①　→②　→③

を繰り返えせば，マイカーがドットを消しながら上にあがっていきます。まとめると，

マイカーを動かす原理

となります。簡単でしょう？

"クモの糸"プログラム

以上の手続きをまとめたのが，**リスト3－10**です。このままでは，少し不満ですね。

① **このままプログラムを走らせると，前の画面が残って見ずらい！**

──OK。プログラム最初に，画面クリアの命令を入れましょう。PCでしたら，

PRINT　CHR$（12）；

です。あなたのマシンでは？

② **マイカーが画面の上端に来ても，そのまま上に行ってしまうので，エラーになる！**

──OK。チェック機能をつけましょう。マイカーが上端に達すると，

タテ座標＝0

になります。したがって

IF　YMYCAR＝0　THEN～

の文を入れて，チェックしましょう。

《第3－16図》マイカーを表示

③ **スピードが早すぎる！**

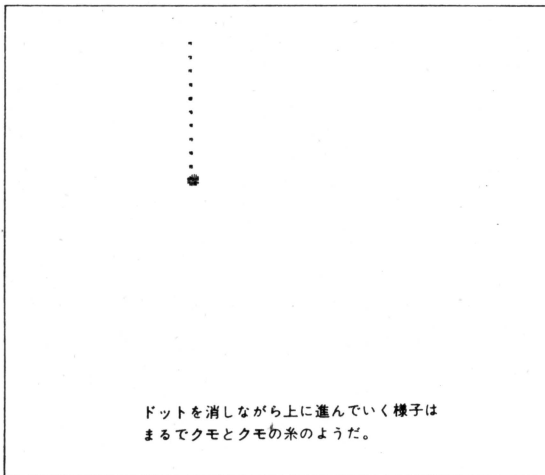
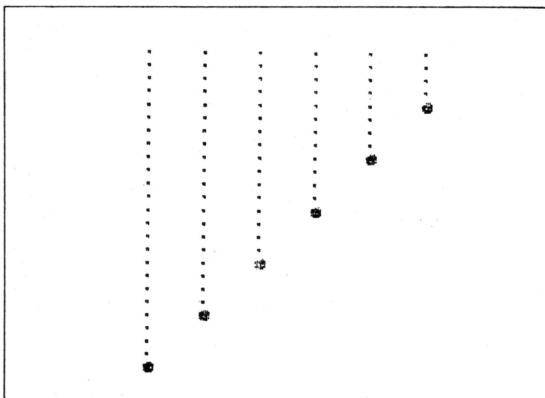
──OK。**タイマー・ルーチン**を途中に入れましょう。タイマー・ルーチンって御存知ですね？

以上の訂正をほどこしたものが，**リスト3－11**です。これを走らせたのが，**第3－17図**です。まるで**クモが糸をのぼっていくみたい**に見えます。そして上までのぼり切ると，ストンと下まで降りてきます。単純なプログラムですが，なかなか面白いですよ。ついでに沢

《リスト3－10》マイカーがドットを消しながら動く

```
110 XMYCAR=10:YMYCAR=24                    'INITIAL
120 FOR Y=0 TO YMYCAR                      'LINE-0 TO LINE-24
130   LOCATE XMYCAR,Y:PRINT "・";
140 NEXT
150 LOCATE XMYCAR,YMCAR:PRINT "●";         'PRINT MYCAR
160 LOCATE XMYCAR,YMCAR:PRINT " ";         'ERASE MYCAR
170   YMYCAR=YMYCA-1                       'MOVE UP MYCAR
900 GOTO 150                               'FOR LOOP
```

《リスト3－11》クモの糸プログラムⅠ（リスト3－10の改良版）

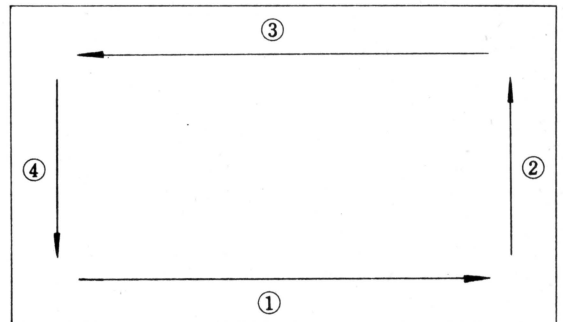
```
100 PRINT CHR$(12)                                    'CLEAR
110 XMYCAR=10:YMYCAR=24                               'INITIAL
120 FOR Y=0 TO YMYCAR                                 'LINE-0 TO LINE-24
130   LOCATE XMYCAR,Y:PRINT "・";
140 NEXT
150 LOCATE XMYCAR,YMCAR:PRINT "●";                    'PRINT MYCAR
155   FOR I=0 TO 30:NEXT                              'TIMEER
160 LOCATE XMYCAR,YMCAR:PRINT " ";                    'ERASE MYCAR
170   YMYCAR=YMYCA-1                                  'MOVE UP MYCAR
900 IF YMYCAR>0 THEN 150 ELSE 110
```

《第３－17図》**マイカーが上に進んでいく**

《第３－18図》**リスト３－12を走らせると**

山のクモをのぼらせてみたのが，**リスト３－12，第３－18図**です。題して

クモとクモの糸

限りなくダサイですね。

マイカー移動のテクニック

以上がこのGAMEの基礎です。いつまでたっても**〝クモの糸〞**ばかりでは，GAMEになりません。そ

《第３－19図》**４方向に番号をつける**

こで次にマイカーを

GAMEエリア上で走らす

ことを考えましょう。

マイカーの位置は，先と同じ

（XMYCAR，YMYCAR）

で決まります。次にマイカーの移動方向ですが，今度は，

四つの方向

があり，先のように単純には済みません。そこで**第３－19図**のように，**各方向に数を対応**させます。さらに

変数MCRS

を導入し，**MCRSに現在のマイカーの方向を記憶**させますとたとえば

MCRS＝2

なら，現在マイカーは上に向かっているところを表わしていますし，これが左に向きを変えれば，

MCRS＝3

になるという具合です。

次に，マイカーの移動――すなわち

XMYCAR，YMYCAR

の値を変えることを考えます。これも

四方向により変位が異なる

ため，正攻法で行くと四つに分けて考えなければなり

《リスト3—12》クモの糸プログラムII（リスト3—11の改良版）

```
1000 '====================================
1010 '  DEMO  [ クモ ノ イト ]
1020 '  `  for INVUITAION FOR GAMING 2-2
1030 '          ( 1982.4.19 )
1040 '                 by K.TSUKAGOSHI
1050 '====================================
1060 '
1070 '--- INITIALIZE ---
1080 WIDTH 40,25:CONSOLE 0,25,0,1              'INITIAL SCREEN
1090 PRINT CHR$(12)                            'CLEAR
1100 '
1110 '--- SET SCREEN ---
1120 DIM X(5),Y(5)
1130 FOR I=0 TO 5
1140   X(I)=I*5+5:Y(I)=24-I*4                  'SET START POINT
1150   FOR Y=0 TO Y(I)
1160     COLOR RND(1)*6+1                      'RADOMIZE COLOR
1170     LOCATE X(I),Y:PRINT "・";              'クモ ノ イト
1180   NEXT
1190   COLOR 7                                 'WHITE
1200   LOCATE X(I),Y(I):PRINT "●";             'SET SPIDERS
1210 NEXT
1220 '
1230 --- LOOP ---
1240 FOR I=0 TO 5
1250   LOCATE X(I),Y(I):PRINT " ";             'ERASE SPIDERS
1260   Y(I)=Y(I)-1                             'MOVE UP
1270     IF Y(I)>=0 THEN 1340                  'CHECK UPER POINT
1280            'ELSE
1290                 FOR Y=0 TO 24
1300                   COLOR RND(1)*6+1
1310                   LOCATE X(I),Y:PRINT "・";
1320                 NEXT
1330                 Y(I)=24
1340   COLOR 7
1350   LOCATE X(I),Y(I):PRINT "●";             'PRINT NEW SPIDERS
1360 NEXT
1370 GOTO 1240
```

ません。たとえば

上に進む：YMYCAR－2

（注，ドット・とドット・の間は2）

左に進む：XMYCAR－2

という具合に。これは面倒ですね？

そこで配列変数

XADD（X），YADD（Y）

を用意し，これに各方向の変位を記憶させておきます。もちろん**配列の番号**は，先に決めた**方向の番号**と対応させます。たとえば，

上への変位（方向＝2）

XADD（2）＝0　；変化ナシ

YADD（2）＝－2；上へ2進める

左への変位（方向＝3）

XADD（3）＝－2；左へ2進める

YADD（3）＝0　；変化ナシ

という具合です。**リスト3—13**が，今月までの完成品です。このリストの

9210～9216：DATA文

は，この変位のDATAを用意したものです。そして170～174行でREAD文を使って配列に読み込んでいます。

ところで，マイカーの移動方向はMCRSが覚えているのでしたね。したがって

XADD（MCRS）

YADD（MCRS）

が現在のマイカーの**変位**を表わしているといえます。ヤヤッコシイなあ！　マア，我慢。

マイカー移動の手順

以上を整理しますと，マイカー移動の手順は，次のようになります。

① **マイカーを消す**

```
LOCATE XMYCAR, YMYCAR
PRINT " ";
```

② **マイカーの位置を進める**

```
XMYCAR=XMYCAR
        +XADD(MCRS)
YMYCAR=YMYCAR
        +YADD(MCRS)
```

③ **新しい位置にマイカーを書く**

```
LOCATE XMYCAR, YMYCAR
PRINT "●";
```

これは繰り返すことにより，マイカーはその進路方向に進んで行きます。ほとんど**〝クモの糸〟**と同じですね？　簡単でしょう？　エッ？　難しい？　スミマセンねェ。BASICに代わって，お詫び申しあげます。ペコリ。

進路変更の分析

まてよ。これだけでよかったかな。

マイカーに今の手順を与えてやると，

MCRS

の方向に走り出す——これはOKですね。でもこのままでは，

その方向に走りっぱなし！

になってしまいますね。もちろんコーナーに着いてもそのまま真っすぐ進んでしまいます。本当はコーナーに着いたら，進路方向左に曲げてやらねばなりません。さあ，どうしましょう？

この問題を解決するには，もう少しMYCARの進路を分析する必要があります。

第3—20図を御覧ください。**MYCARは，GAMEエリア上の位置により次の11の進み方**があります。

① コースの途中にいるときは，そのまま前進する。

② 進路を左に変える（MCRS＝3　にする）。

② 進路を下に変える（MCRS＝4　にする）。

④ 進路を右に変える（MCRS＝1　にする）。

⑤ 進路を上に変える（MCRS＝2　にする）。

⑥ キースキャン。もし[4]のキー（**第3—12図**参照）が押されていたら，マイカーのコースを一つ内側に変える。キーが押されていなければ，そのまま前進させる。

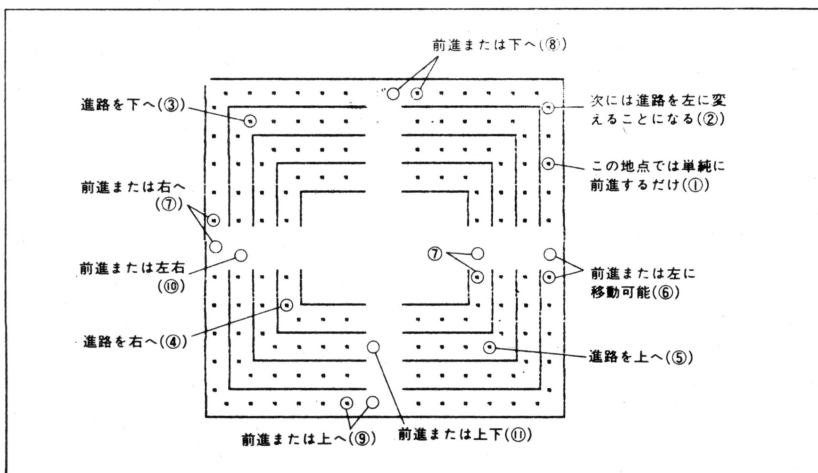
⑦ キースキャン。[6]のキー。

⑧ キースキャン。[2]のキー。

⑨ キースキャン。[8]のキー。

⑩ キースキャン。[4]または[6]のキー。これは，三方向に進路を変えられる場合で，[4]が押されていれば内側のコースに，[6]のキーが押されていれば外側のコースにそれぞれ進路変更をします。もし何もキーが押されていなければ，そのまま前進させます。

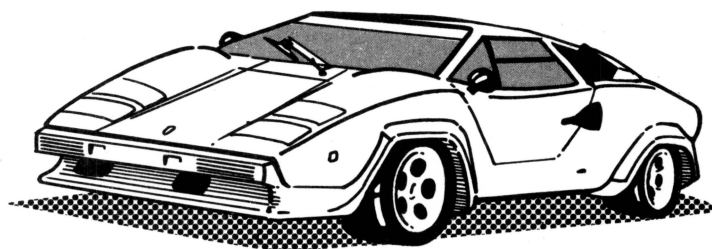
⑪ キースキャン。[2]または[8]のキー。

さあ，マイカーを動かそうとすると，これだけのチェックをしなければならないのです。あなたは，これをどうやってプログラム化しますか？

《第3—20図》**車の進路変更（11のケース）**

《リスト3—13》BOMBERプログラムリスト(第2章)

```
10 '♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
12 '♥  BOMBER for 'INVITATION FOR GAMING'2-2 ♥
14 '♥                 << 82.2.15-X.XX >>      ♥
16 '♥                        by K.TUKAGOSHI ♥
18 '♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
19 '
20 '—— VARIABLE ——
22 'HISCR              :ﾊｲ ｽｺｱ
24 'LEFT               :COUNTER OF LEFT CAR
26 'SCR                :ｽｺｱ
28 'XMYCAR,YMYCAR      :LOCATE MYCAR
30 'C1,C2              :MYCAR ｺｰｽ & ｼﾝﾛ
32 'MCRS               :DIRECTION OF MYCAR      (1=ﾐｷﾞ,2=ｳｴ,3=ﾋﾀﾞﾘ,4=ｼﾀ)
34 'CMEMO              :MEMO of MCRS(C1,C2) OR RCRS(C3,C4)
58 '
59 '—— DIMENSION ——
60 'CRS(ｺｰｽ,ｷｮﾘ)       :ﾜｸ =0,1,2,3 (0=OUTSIDE,3=INSIDE)
61 '                   :ｼﾝﾛ=0      ｾﾞﾝｼﾝ  [ﾂｷﾞ ﾊ ﾏｲｶｰ ﾉ ﾊﾞｱｲ]
62 '                   :    =1-4   ﾎｳｺｳ ﾃﾝｶﾝ       (1=ﾐｷﾞ,2=ｳｴ,3=ﾋﾀﾞﾘ,4=ｼﾀ)
63 '                   :    =5-8   ｷｰｽｷｬﾝ 1 ﾎｳｺｳ (5=ﾐｷﾞ,6=ｳｴ,7=ﾋﾀﾞﾘ,8=ｼﾀ)
64 '                   :    =9,10  ｷｰｽｷｬﾝ 2 ﾎｳｺｳ (9=ｻﾕｳ,10=ｼﾞｮｳｹﾞ)
65 '                   :    =99    ｼﾝﾛ ｶｳﾝﾀｰ=0 ﾍ
66 'XADD(X),YADD(Y)    :ﾍﾝｲ X,Y
99 '
100 '▬▬▬ MAIN ROUTINE ▬▬▬
110 '—— COLD START ——
115 DEFINT A-Z                                   'SET INTEGER
120 HISCR=0
150 DIM CRS(3,50),ADD(4,4)
160   RESTORE 9110                               'READ CRS(3,50)
162   FOR I=0 TO 3
164     FOR J=0 TO 50
166       READ CRS(I,J)
168   NEXT J,I
170   FOR I=1 TO 4                               'READ XADD(4),YADD(4)
172     READ XADD(I),YADD(I)
174   NEXT
199 '
200 '—— HOT  START ——
210 SCR=0:LEFT=3
220 XMYCAR=29:YMYCAR=22:C1=0:C2=0 :MCRS=2        'RESET MYCAR
250 WIDTH 40,25:CONSOLE 0,25,0,1:COLOR 5,32,0:PRINT CHR$(12);
260 GOSUB 1000                                   'ｶﾞﾒﾝ
270   COLOR 7:LOCATE XMYCAR,YMYCAR:PRINT "●";
280 GOSUB 1600                                   'MOVE MYCAR
900 GOTO 280                                     'MAIN END
999 '
1000 '▬▬▬ SUB ROUTINE ▬▬▬
1001 '—— ｶﾞﾒﾝ ——
1010 COLOR 5:PRINT CHR$(12);                     'PRINT
1020 FOR Y=1 TO 23
1022   X=(Y MOD 2)+1
1024   FOR X=X TO X+28 STEP 2
1026     LOCATE X,Y:PRINT "･";
1028 NEXT X,Y
1030 FOR I=0 TO 8 STEP 2                         'PRINT RECTANGLE
1031   IF (I MOD 8)=0 THEN COLOR 1 ELSE COLOR 5
1032   LOCATE I,I:PRINT "┌";
1033   Y1=I:Y2=24-I
1034   FOR X=I+1 TO 29-I
1036     LOCATE X,Y1:PRINT "─";:LOCATE X,Y2:PRINT "─";
1038   NEXT
1039   LOCATE X,Y1:PRINT "┐";:LOCATE X,Y2:PRINT "┘";
1040   X1=I:X2=30-I
1042   FOR Y=I+1 TO 23-I
1044     LOCATE X1,Y:PRINT "|";:LOCATE X2,Y:PRINT "|";
1046   NEXT
1048   LOCATE X1,Y:PRINT "└";
1050 NEXT
1060 X1=9 :X2=21:Y1=9 :Y2=15:GOSUB 1100          'CROSS OUT
1062 X1=14:X2=16:Y1=1 :Y2=23:GOSUB 1100
1064 X1=1 :X2=29:Y1=11:Y2=13:GOSUB 1100
1070 RESTORE 9020                                'PRINT TITLE
1071 FOR Y=1 TO 24
1072   READ I$:COLOR 2:LOCATE 31,Y:PRINT "◣";:COLOR 6:PRINT I$;
1073   COLOR 2:PRINT "◢";
1074 NEXT
```

```
1080 COLOR 2:LOCATE 10,10:PRINT "LEFT:    ";:GOSUB 1200 'PRINT MESSAGE
1082 COLOR 4:LOCATE 13,12:PRINT    "SCORE";:GOSUB 1300
1084 COLOR 4:LOCATE 10,14:PRINT "HI-SCORE";:GOSUB 1400
1090 RETURN
1099 '
1100 '—— CLEAR RECTANGLE ——————
1101 ' ——— PARA IN:X1,X2,Y1,Y2 ——
1110 FOR Y=Y1 TO Y2
1120   FOR X=X1 TO X2
1130     LOCATE X,Y:PRINT " ";
1140 NEXT X,Y:RETURN
1199 '
1200 '—— PRINT LEFT ——
1210 IF LEFT=0 THEN RETURN                        'LEFT=0 THEN ヒョウジ セズ
1220 FOR X=15 TO 14+LEFT
1230   COLOR 7:LOCATE X,10:PRINT "●";
1240 NEXT:RETURN
1299 '
1300 '—— PRINT SCORE ——
1310 COLOR 7:LOCATE 15,13:PRINT RIGHT$("0000"+HEX$(SCR),5):RETURN
1399 '
1400 '—— PRINT HI-SCORE ——
1410 COLOR 7:LOCATE 15,15:PRINT RIGHT$("0000"+HEX$(HISCR),5):RETURN
1499 '
1500 '—— CHANGE DIRECTION MYCAR ——
1510 XMYCAR=XMYCAR-1:YMYCAR=YMYCAR+1:MCRS=1:RETURN     'FOR RIGHT
1520 XMYCAR=XMYCAR+1:YMYCAR=YMYCAR+1:MCRS=2:RETURN     'FOR UP
1530 XMYCAR=XMYCAR+1:YMYCAR=YMYCAR-1:MCRS=3:RETURN     'FOR LEFT
1540 XMYCAR=XMYCAR-1:YMYCAR=YMYCAR-1:MCRS=4:RETURN     'FOR DOWN
1599 '
1600 '—— MOVE MYCAR ——
1610 COLOR 5:LOCATE XMYCAR,YMYCAR:PRINT " ";    'ERASE MYCAR
1620   CMEMO=CRS(C1,C2)                         'CMEMO=シンロ
1630   IF CMEMO=99 THEN C2=0:GOTO 1620 ELSE C2=C2+1
1640   ON CMEMO GOSUB 1510,1520,1530,1540,2100,2200,2300,2400,2500,2600
1650   XMYCAR=XMYCAR+XADD(MCRS):YMYCAR=YMYCAR+YADD(MCRS)
1660 COLOR 7:LOCATE XMYCAR,YMYCAR:PRINT "●";    'PRINT MYCAR
1690 RETURN
1699 '
2100 '—— GO RIGHT? MYCAR ——
2110 CMEMO=0
2190 RETURN
2200 '—— GO UP? MYCAR ——
2210 CMEMO=0
2290 RETURN
2300 '—— GO LEFT? MYCAR ——
2310 CMEMO=0
2390 RETURN
2400 '—— GO DOWN? MYCAR ——
2410 CMEMO=0
2490 RETURN
2500 '—— GO LEFT OR RIGHT? MYCAR ——
2510 CMEMO=0
2590 RETURN
2600 '—— GO UP OR DOWN? MYCAR ——
2610 CMEMO=0
2690 RETURN
2700 '—— GO RIGHT? MYCAR ——
2710 CMEMO=0
2790 RETURN
2800 '—— GO UP? MYCAR ——
2810 CMEMO=0
2890 RETURN
2900 '—— GO LEFT? MYCAR ——
2910 CMEMO=0
2990 RETURN
3000 '—— GO DOWN? MYCAR ——
3010 CMEMO=0
3090 RETURN
3100 '—— GO LEFT OR RIGHT? MYCAR ——
3110 CMEMO=0
3190 RETURN
3200 '—— GO UP OR DOWN? MYCAR ——
3210 CMEMO=0
3290 RETURN
8999 '
```

```
9000 '▬▬▬ DATA AREA ▬▬▬
9010 '—— TITLE DATA ——
9020  DATA "█▔▔◣"
9022  DATA "█▁▁◤"
9024  DATA "█  ◣"
9026  DATA "█▁▁◤"
9028  DATA "◢▔▔◣"
9030  DATA "█  █"
9032  DATA "█  █"
9034  DATA "◥▁▁◤"
9036  DATA "◣  ◢"
9038  DATA "█◣▁◢█"
9040  DATA "█ ▔ █"
9042  DATA "█  █"
9044  DATA "█▔▔◣"
9046  DATA "█▁▁◤"
9048  DATA "█  ◣"
9050  DATA "█▁▁◤"
9052  DATA "█▔▔◥"
9054  DATA "█▁  "
9056  DATA "█   "
9058  DATA "█▁▁◢"
9060  DATA "█▔▔◣"
9062  DATA "█▁▁◤"
9064  DATA "█ ◥ "
9066  DATA "█▁ ◥"
9098 '
9099 '—— COURSE DATA ——
9110 DATA 0,0,0,0,7,7,0,0,0,0,3,0,0,0,0,0,8,8,8,0,0,0,0,0,4:   'COURSE-0
9112 DATA 0,0,0,0,5,5,0,0,0,0,1,0,0,0,0,0,6,6,6,0,0,0,0,0,2
9114 DATA 99
9116 DATA 0,0,0,9,9,0,0,0,3,0,0,0,0,10,10,10,0,0,0,0,4:        'COURSE-1
9118 DATA 0,0,0,9,9,0,0,0,1,0,0,0,0,10,10,10,0,0,0,0,2
9120 DATA 99,0,0,0,0,0,0,0,0
9122 DATA 0,0,9,9,0,0,3,0,0,0,10,10,10,0,0,0,4:                'COURSE-2
9124 DATA 0,0,9,9,0,0,1,0,0,0,10,10,10,0,0,0,2
9126 DATA 99,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
9128 DATA 0,5,5,0,3,0,0,6,6,6,0,0,4:                           'COURSE-3
9130 DATA 0,7,7,0,1,0,0,8,8,8,0,0,2
9132 DATA 99,0,0,0,0,0,0,0,0
9134 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
9139 '
9200 '—— ADD DATA ——
9210 DATA +2, 0:                          'RIGHT
9212 DATA  0,-2:                          'UP
9214 DATA -2, 0:                          'LEFT
9216 DATA  0,+2:                          'DOWN
```

配列によるシミュレート

この進路変更チェックのために私は，

配列変数CRS（3，50）

を用意しました。これは**GAMEエリアの各道路状況を覚えておく**ためのもので，

第1パラメータ：コース番号

第2パラメータ：基準点からの距離

配列の値：その位置の道路状況

を表わしています。いわば**GAMEエリアの全道路をシミュレート**しているわけです。

配列変数CRSについて，まずパラメータの意味から説明していきます。

第3—21図のように，GAMEエリアは

四つのコース

からできあがっています。この四つのコースに外側から番号（0オリジン）をつけます。たとえば

一番外側のコース＝0

一番中側のコース＝3

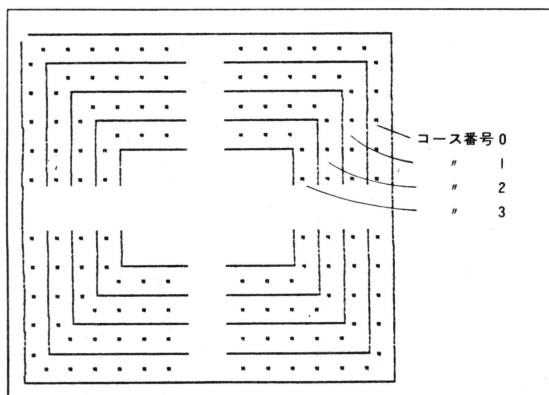
のように。そうです。配列CRSの1番目の添字は，このコース番号のことです。

次に**第3—22図**を御覧ください。各コースの基準点をこの図のように定めます。そして各ドット・の位置を基準点から左まわりに数えて決めます（これも0オリジンです）。たとえば

A点＝3（コース番号0）

B点＝7（コース番号1）

C点＝8（コース番号3）

のように。そしてこの基準点からの距離を配列CRS2番目の添字に取るのです。

さあ，これで配列CRSの二つのパラメータの意味はお分りいただけたと思います。ここで注意していただきたいのは，

GAMEエリアの全ての位置は，

二つのパラメータで決定される！

ということです。すなわち，GAMEエリアの位置が決まればコース番号と基準点から距離は定まります。また逆に二つのパラメータの値がわかれば，それがコース番号と基準点からの距離を表わすことになり，GAMEエリアの位置が決まります。以上のことから

GAME上のエリアと

配列CRSは1対1対応している

といえるわけです。

進路状況コードの導入

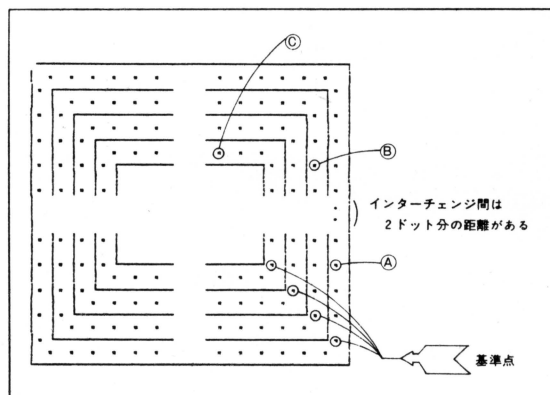
配列の良いところは，それに値を記憶できるということです。そこで各配列の値として先に見た

11の進路変更のパターン

を記憶させておくことにします。その値は，次のように決めることにします。

0——前進
1——右に進路変更
2——上に進路変更
3——左に進路変更
4——下に進路変更
5——キースキャン，右チェック
6——キースキャン，上チェック
7——キースキャン，左チェック
8——キースキャン，下チェック
9——キースキャン，左右のチェック
10——キースキャン，上下のチェック

これにより，**配列CRSは**

《第3—21図》**四つのコース**

《第3—22図》**各コースの基準点**

〈入力〉：ドット・の位置

〈出力〉：進路状況コード（0～10）

の入出力を持つ2変数1価関数（変換器）の役目を持つことになります。

以上で準備はすべて整いましたから，次にその具体的な使い方を説明致します。

CRS上を走らせる

プログラムを走らせると，マイカーはGAMEエリア上を元気良く走って行きます。しかしプログラムから見るとマイカーは，

配列変数CRS上を走っている

に過ぎないのです。

マイカーは，GAMEエリア上の位置を表わすものとして，

XMYCAR，YMYCAR

という二つの変数を持っています。と同時に配列CRS上の位置を表わす変数として

C1：コース番号を記憶

C2：基準点からの距離を記憶

を持たせます。C1，C2の初期値は

C1＝0：C2＝0

です。これにより，マイカーの移動手順は次のようになります。

① （XMYCAR，YMYCAR）で示されるGAMEエリア上のマイカーを消去する。

② CRS（C1，C2）の値により，現在の進路状況を調べる。たとえば，

CRS（C1，C2）＝0

ならそのまま直進だし，

CRS（C1，C2）＝4

なら下に向きを変えるという具合に。

③ C2は単純に一つ大きくする。これは配列上を一つ前進することになります。

④ ②の進路状況により，それぞれの処理ルーチンに制御を移す。

コース末の処理

これで第2章のリストは，全部読めると思います。しかし，二つだけ疑問が残ると思いますので，その点を説明しておきましょう。

その疑問とは，

① コースの一周をどうやって知る。？

② 各コースの一周の長さは異なるが，不都合は生じないか？

の2点です。さあ，どうでしょうか？

マイカーはCRS上を走りながら

C2＝C2＋1

と配列の添字を大きくしていきます。ですからこのままでは当然エラーとなります。

私のプログラムでは，この2点を解決するために

ENDマーク＝99

を用意しています。つまり各コースを一周すると，

CRS（C1，C2）＝99 ――Ⓐ

になるようにしているのです。そして，進路状況がⒶのようになったところで

C2＝0

としてコースの最初に戻してやっているのです。するとマイカーはまたコースの基準点から走り出すことになるわけです。

以上のENDマークに注意して**リスト3―13**の9110行からのDATAを御覧になってください。各コース末にENDマークが置かれていますね。コースは内側になる程短くなりますから，ENDマークのあとにダミーのDATAが入っています。

どうですか？ このENDマークを導入することで上記の2点が解決されましたね。

第2章のまとめ

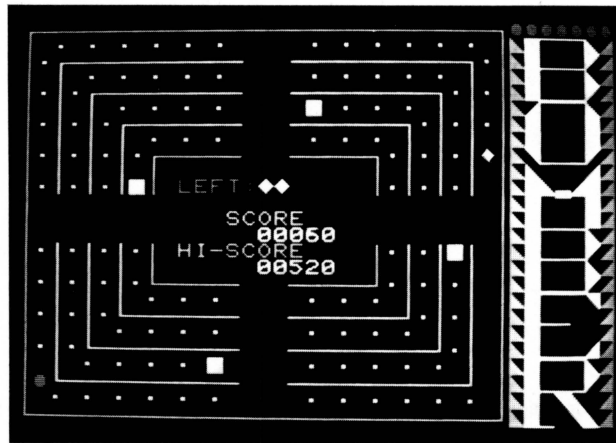
第2章では，**マイカーの移動処理**について

GAMEで要求される仕様

を詳しく分析してみました。そして第1章と同様にその**考え方**を中心に話しを進め，**プログラミングそのもの**にはタッチしませんでした。具体的なプログラミングについては**リスト3―13**を御覧いただき，あなた自身の手によりあなたのマシン上で実現してみてください。そのとき**リスト3―13**に現われているPCのハードに係る命令については無視してください。**リスト3―13**を走らせると，マイカーがコース0上を周回する様子が再現されるはずです。

第3章

リアルタイムゲーム「BOMBER」に挑戦

SKIPマークの導入

はじめに

ひとつ　必死でGAME作り
ふたつ　不可解なプログラミング
みっつ　見つけたアルゴリズム
よっつ　喜べ完成だ
いつつ　いつしかバグの山
（尻魂バレーのプログラマー）

プログラムの構造を考える

第2章の**マイカーの移動**，うまくいきましたか？

その**原理**，わかりました？　エッ，プログラムの説明がまだだ？　そうでした，そうでした。

それでは，最初に第2章で作ったプログラムを見ておきましょう。マイカーの動きを中心に。

まずは第3—23図を御覧ください。ここに，

プログラムの一般的構造図

を示しておきました。プログラムをこのような形にまとめると，全体が**非常にスッキリした見やすい構造**になります。

最初の“**前処理**”とは，一般に“**初期設定**”と呼ばれています。狭義には，変数の初期化のことを指します。その主旨は，処理の本体に必要な**前処理**をプログラムのあちこちに散らすのでなく，前の方の一個所に集めておきましょうということです。

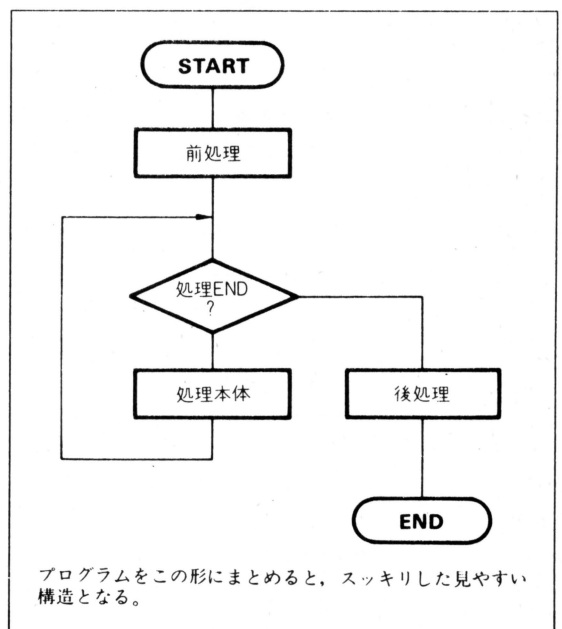
処理の本体は，

DO ～ UNTIL形

を用いてEND条件の判定を**モジュールの入口**でおこないます。したがって極端な場合は，

処理本体が一度も実行されない

ということも起こるわけです。このことは重要で，すべてのループは，

DO ～ UNTIL形

《第3—23図》**プログラムの一般的構造**

で記述可能なことを物語っています。

END条件に達すると，**“後処理”ルーチン**に抜けます。TV画面を通してのリアルタイム処理が多いマイコンの場合，とくにこの“後処理”が不要の場合があります。

以上のようなスッキリした構造にプログラムをまとめるには，**設計方針（態度）**が影響を与えます。すなわち

ボトム・アップ ――――×

ではなく，

トップ・ダウン ――――○

の設計態度でのぞむべきです。それが

構造化プログラミング

(structured programming)

の基本です。

準備

マイカーの移動についても，**初期設定が必要**です。

変数は，配列変数と単純変数に分かれます。配列変数は，

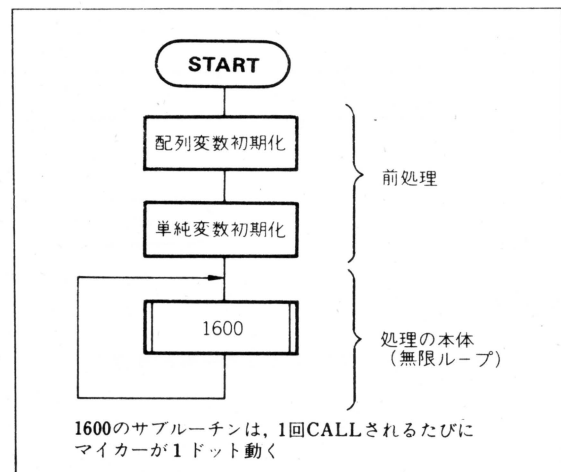
CRS：道路状況の記憶

ADD：次の位置までの変位

の二つでしたね。150行で定義し，162～174行で必要なDATAを読み込みます。160行の

RESTORE

は，READ文でDATAを読むときのポインタを変更するものです。DATAの順序を読み込む順序に並べておけば，一般には不要です。

マイカー関係の単純変数は，220行で初期化しています。そして270行で初期位置にプリントします。

《第3—24図》**マイカー移動の処理構造**

マイカー移動ルーチンの分析

次は，**“マイカー移動”の本体**にかかわる部分になります。そのメインルーチンは，わずか2行です。

280行：1600のサブルーチンをCALL

900行：280行に戻す

すなわちこのプログラムは，**無限ループ**となっています（第3—24図）。そして処理の中身は，すべて

サブルーチン1600行

に依託しています。1600行からのサブルーチンは，1回CALLされるたびに“マイカー”を1ドット動かす処理をおこなうことになります。

さて，それではその**1600行からのサブルーチン**を見てみましょう。いつもフローチャートばかりではワン・パターンなので，今度は

NSチャート

(Nassi Shneiderman Chart)

にまとめてみました（第3—25図）。

まず1610行で，“マイカー”を消去します。と同時に**ドット・も消去される**ことに注意してください。次に**道路の進路状況**を調べます。第2章で分析したように

CRS（C1，C2）

が，道路の進路状況を表わしています。この値は，

0～10　または　99

の12種類でしたね。その値を

変数CMEMO

に記憶します。

さて，ここでCMEMOの値が99（ENDマーク）に達したかチェックします。そして，次の二つのケースに分かれます（1630行）。

① **CMEMO＝99のとき**

マイカーは，コースを一周したわけです。したがって**距離を基準点に戻し**

C2＝0

もう一度CMEMOに新しい進路状況を入れ直してやります。

CMEMO＝CRS（C1，C2）

② **CMEMO＝0～10のとき**

単純に基準点からの距離を一つ増やしてやります。

C2＝C2＋1

これでCMEMOには，0～10の11種類の進路が入ったことになりますので，1640行の

ON ～ GOSUB

により**それぞれの処理ルーチンをCALL**してやりま

す。それは，大きく次の3種類に分れます。

① **CMEMO＝0のとき ON～GOSUBにはひっかかりません。**このときはどのサブルーチンもCALLされず，直接下の1650行に抜けます。すなわち単純な直進になります。

② **CMEMO＝1～4のとき**

　ちょうどコーナーにさしかかったところにあたります？それぞれ**カーブを曲るサブルーチン**に飛び込みます（1510～1540行)。この部分については，あとでもう一度補足することにします。

（行番号） 1600
（内　容） マイカーを1ドット動かす

ENTRY
マイカーを消去する
CMEMO←進路番号　CRS(C1, C2)
Y　CMEMO＝99？　N
C2＝0 ／ CMEMO←新しい進路番号 ‖ 基準点からの距離C2を1つ増やす C2＝C2＋1
CMEMOの値により，それぞれの処理をおこなう ON CMEMO GOSUB～
1: 1510 ‖ 2: 1520 ‖ 3: 1530 ‖ 4: 1540 ‖ 5: 2100 ‖ 6: 2200 ‖ 7: 2300 ‖ 8: 2400 ‖ 9: 2500 ‖ 10: 2600
(XMYCAR, YMYCAR)の更新
新しい位置にマイカーを表示する
RETURN

《第3—25図》**NSチャート：MOVE　MYCAR**

③ **CMEMO＝5～10　のとき**

　キースキャン処理です。この部分は，第4章以降に作ることになりますから，現在は何もなかったことにして，

CMEMO＝0

にした上で戻ってきます。

　次に1650行で，“マイカー”の新しい位置

（XMYCAR，YMYCAR）

を計算します。

MCRS

が現在の進路方向，また

XADD（MCRS）：X方向
YADD（MCRS）：Y方向

が各方向の変位を表わしていますから，1650行の計算で新しい位置が得られるわけです。この式は，**納得いくまで良く味わって**ください。なかなかスマートでしょう？

　そして1660行で新しい位置に“マイカー”を表示してやります。

コーナーにおける盲点

　以上が，“マイカー移動”についての

メインルーチン
サブルーチン

の解析です。ここで保留しておいた

コーナーの曲り方

について補足しておきましょう。

　コーナーでカーブを切るとき，

上，下，左，右

の四つの曲り方があります。ここでは

直進——→左に曲がる

場合を例にとり，説明しておくことにします。

　第3—26図①のようにコースを直進してきて，**A地点**に達したとします。すると次は，**B地点**に進めてやれば良いことになります。さてA地点においては，

CRS（C1，C2）＝3

になりますから1620行で

CMEMO＝3

になります。すると次の1640行でGOSUBの3番目，1530行のサブルーチンに飛び込むことになります。

　そこで1530行を見てみましょう。

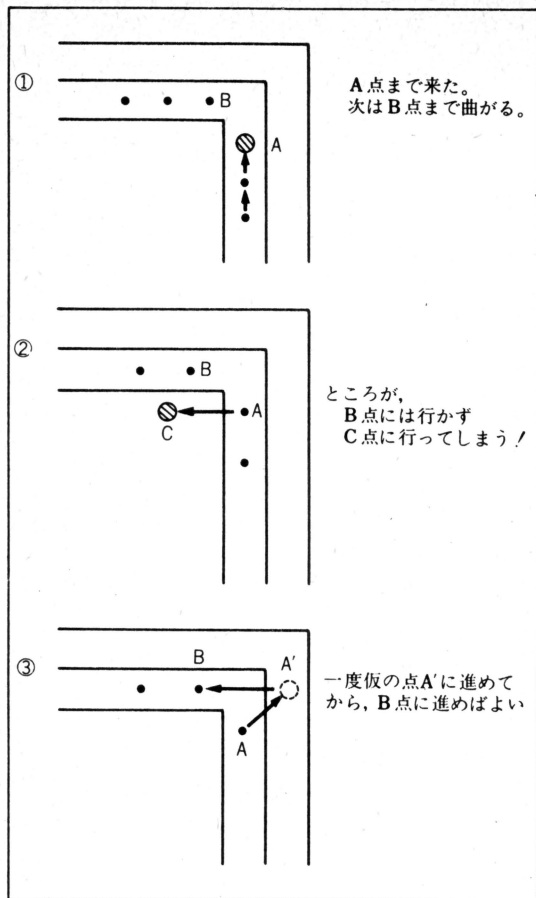
MCRS＝3

はわかりますね？　今度は進路方向が左になるからです。それではその前の

（XMYCAR，YMYCAR）

の値をいじくっているのは，何をしているのでしょうか？

　ここで単純に1530行を

MCRS＝3

《第３―26図》**御注意！コーナー**

だけ実行してRETURNしたとします。すると

　　1650行，1660行

でマイカーが次の位置（この場合，左に進む）に進められます。これを図で見ると（**第３―26図②**），A地点から左に進むことになり，**B地点にはいかず，C地点に進む**ことになります。これは少しおかしいですね。

そこで**第３―26図③**のように，"マイカー"の位置をA地点からA'地点に調整してやります。もちろんこれは

　　（XMYCAR，YMYCAR）

の変数の値を調整してやるだけで，実際に"マイカー"をそこに表示するわけではありません。こうしてやれば，次の1650～1660行でちゃんとB地点に進んでくれるわけです。1530行における最初の計算は，実はこの調整をやっているところだったのです。

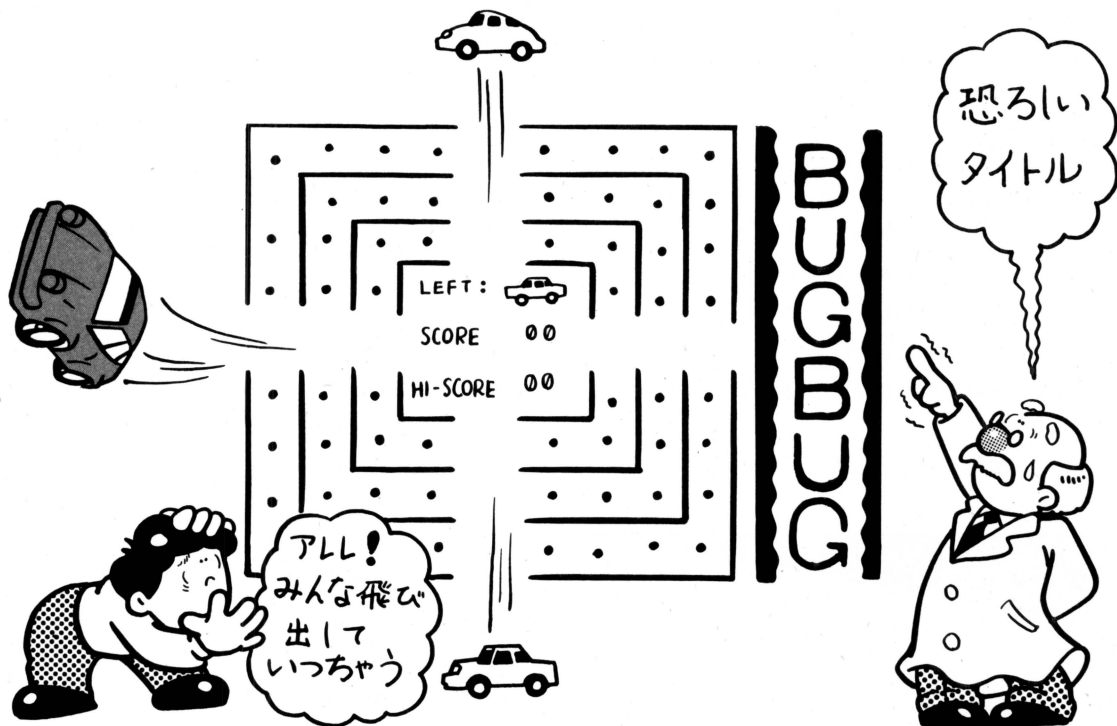
いかがでしたか？

　　"BOMBER"のように動きのあるGAMEでは，以上のように常に細いところにも細心の注意を払ってプログラミングすることが要求されます。さもないと，

　　登場人物がおかしな動き？

をすることになり，

「バグだ！　バグだ！」

とおこられることになります。

“レッド・カーの考察

次の目標は，

対向車――REDCAR

の移動です。

我々は，すでに**“マイカーの移動”に成功**しています。したがって**“レッド・カーの移動”**についても基本的な考え方はわかっていることになります。そこで

マイカーとレッド・カー

の動きで異なる点を列記してみましょう。

① まわる向きが逆

② ドット・は消さない

③ インターチェンジでの扱いが異なる

プログラム上で異なる扱いを受けるのは，以上3点でしょう。それぞれひとくふう必要そうですね？ それではまずこの3点について簡単に考察してみましょう。

① **まわる向きについて**

“マイカー”については，各位置の進路状況を表わすテーブル（配列）を準備しました。“レッド・カー”にも同じようなテーブルを作れば良いですね？ でもわざわざ似たようなテーブルを作るのは面倒だし。できれば**“マイカー”のテーブルをそのまま利用**したいと思いませんか？ ただし，そのまま利用してしまうと

“レッド・カー”も同じ左まわり

になってしまいます。どうしましょうか？

② **ドット・を消さない**

これもひとくふう必要なようです。なぜなら“レッドカー”が進むたびに

ドット・があるのか？

ドット・がないのか？

判定しなければなりません。少々難しそうですね？

③ **インターチェンジでの扱い**

これは見た目にも明らかに異なりますね？ “マイカー”でしたらインターチェンジでキースキャンを行い，**キーの押し具合**でコースを変えます。ところが“レッド・カー”はキースキャンではなく，**“マイカー”の位置**でコースを変えることになります。

ただし③については，とりあえずは考える必要はないでしょう。なにせ我々はまだ

コースを変える問題

を扱っていませんから。これらは第4章以降の課題です。

DATA構造の変更

それでは以上の問題をひとつひとつ解決して行きましょう。

最初に第2章と比べ変更した点を申し上げます。それは，MYCARとREDCARのキャラクタで，

MYCAR―――◆

REDCAR――●

に変更しました。これは**視覚的な気分によるもの**です。◆より●の方が大きく見えるため，恐怖感が増します。私はプログラムの進行とともに，どんどん気分でプログラムを変えていくクセがあります。悪しからず。

さて次に進路状況を表わすテーブル（配列CRS）ですが，やはり**MYCARとREDCAR**で

共 有 化

させることにしましょう。そのためにはDATA構造は少し変えた方が良いでしょう。第3章のリストの9110行～9132行を御覧ください。変更点は次の通りです。

① **すべてを2ケタにした**

これは単なる視覚的な問題です。たとえば

4も**04**も変数の値としては同じ

です。

② **ENDマーク99の扱いを変えた**

これは，次の2点の理由によります。

- 最後にENDマークがあると，逆まわりがやりにくい。
- コースによって長さが異なると，**進路変更にともない縮尺・拡大処理**をしなければならず，プログラムが難しくなる。

以上2点により第3章のプログラムでは，

ENDマークを廃止し，

SKIPマークを導入

しました。こちらの方が少しわかりにくいかもしれません。しかし，ENDマークを理解されたあなたでしたら，十分にこなせるでしょう。

SKIPマークとは?

それでは，SKIPマークを説明致します。

まず，第2章で**なぜENDマークが導入されたか**を考えてみましょう。それは，

コースによりドット・の数が異なる

からでしたね（**第3－27図**）。内側のコースが外側のコースよりドットが少ないのは当り前です（2次元の世界

では）。しかしコースによって長さの扱いが異なるとプログラム処理が面倒になります。そこで

各コースの長さはそのままにし，
ＥＮＤマークでコース末を知る！

という方法を取ったのです。

ところがＥＮＤマークを用いると，前節で見たように二つの不都合が生じます。そこでＥＮＤマークは用いず，かつ各コースの長さを等しくする手段として

ＳＫＩＰマーク

を登場させるわけです。

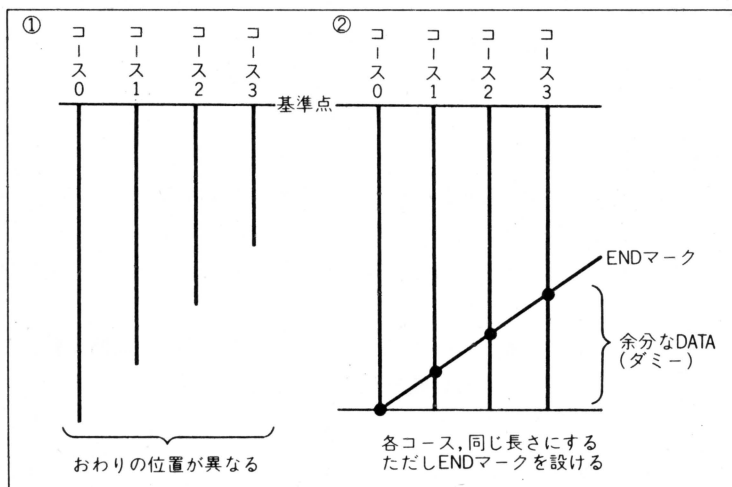
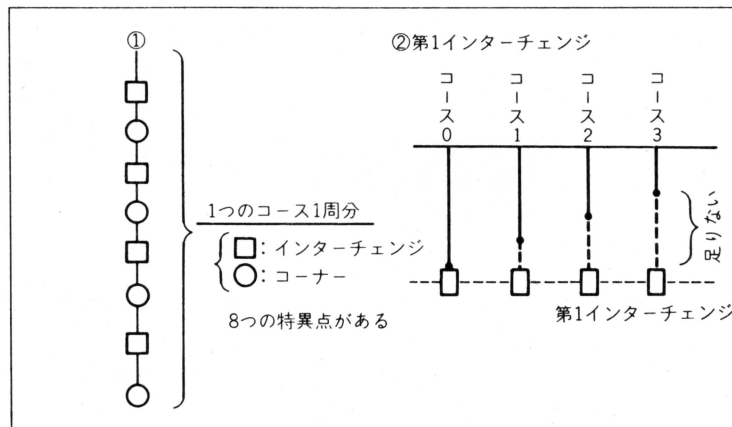
第３－28図①を御覧ください。これが一つのコースを表わしているとします。一つのコースには，

八つの特異点

が存在します。それは

四つの**インターチェンジ**
四つの**コーナー**

から構成されています。

さて，この**八つの特異点までの距離を**

《第３－27図》ＥＮＤマークの役割

《第３－28図》特異点間の距離をそろえる

各コース同じにする

ことを考えてみます。**第３－28図②**を御覧ください。基準点から第１インターチェンジまでを模型化してあります。当然，各コースの長さが異なりますから**コース１～コース３ではデータが不足**します。ところでこの不足するところに仮のデータを入れたらどうなるでしょうか？　各コース，

基準点からインターチェンジまでの
距離が同じ！

になりますね？　そのために入れるのが

ＳＫＩＰマーク

です。もう一度リストの9110行～9132行を御覧ください。ところどころに存在する９９がＳＫＩＰマークです。

続いて1600行～1690行の**“マイカー移動ルーチン”**を御覧ください。ＳＫＩＰマーク導入により，次の３点の変更が生じています。

①　ＳＫＩＰマークのスキップ

ＳＫＩＰマークを見つけると，そこをスキップさせる（1625行）。蛇足ながらＢＡＳＩＣのプログラムでは途中に自由にブランク（空白）を入れることができます。これは，ＢＡＳＩＣインタプリタが，**ブランクをＳＫＩＰマークとしてスキップさせるサブルーチン**を持っているから可能となっています。

②　一周終了の判定

今度はＥＮＤマークがありません。どうしましょう？　簡単です。今度は各コースの長さが同じです。したがって

Ｃ２＝４９

を越えたら一周終了です（1630行）。

③　配列の長さの変更

②によりＥＮＤマークがなくなりましたから，配列ＣＲＳの長さが一つ小さくなります。配列の定義（150行），データの読み込み（164行）を変更しましょう。

レッド・カー”に挑戦

SKIPマークの使い方がわかりましたら，もう一度新しい“マイカー移動ルーチン”を解析してみてください。十分に理解できると思います。

次に問題の**“レッド・カー”の移動**にとりかかりましょう。

画面上の位置：XRED，YRED

配列上の位置 { **コース**：C3
基準点からの距離：C4 }

現在の進行方向：RCRS

以上の変数を導入します。変数の役割は“マイカーの移動ルーチン”と同じです。そこで両者の違いを説明しておきましょう。

① **初期設定が異なる**

220行と230行を比べてください。そして**第3－29図**を御覧ください。違いは明らかですね？

② **C4は減少する**

C4は“レッド・カー”が進行するにしたがって減少します。なぜなら“レッド・カー”は右まわりですから**進めば進むほど基準点に近づく**からです。1630行と1715行を比較してみてください。

③ **一周の判定が異なる**

“マイカー”のC2は増えますから

C2＞49

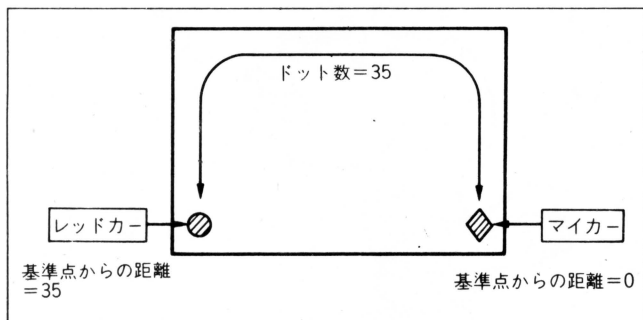
で一周終りですが，“レッド・カー”のC4は減りますから

C4＜0

で一周終りとなります。

④ **配列CRSの扱いが異なる**

たとえば**第3－30図**のように右上コーナーで考えてみましょう。配列CRSで得られるDATAは，図のようになっています。

MYCAR：A点で**左折**

REDCAR：B点で**右折**

しなければなりません。ところが“マイカー”ならA点の**データ4で左折をキャッチ**できますが，“レッド・カー”の場合，B点のデータではキャッチできません。困りましたねェ。

そこで次のように考えることにしましょう。これで解決できますよ。

- “レッド・カー”では，**CRSのデータを読むとき，一歩先のデータを読む**
- データの**解決の仕方を右まわりに変更**する。たとえば**第3－30図**のように

4……MYCARでは**左折**
REDCARでは**右折**

に解釈します。

最後の難問――軌跡

“レッド・カー”の移動については，もう一点，重大な難問が残っていましたね？　自分の軌跡に

ドット・を残すか

ドット・を残さないか

の判定です。

ここでは，ドット・の残り状態を表わす

配列　DOT（コース，位置）

を登場させることで解決してみました。その手順は次の通りです。

① 最初，すべてのDOTの値は1にしておく。**1**はそこにドット・があることを表わします（205行）。

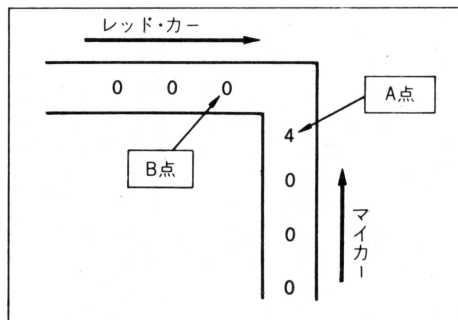
② “マイカー”がドット・を消すと

DOT（C1，C2）＝0

とする。**0**は，そこにドット・がないことを示します（1622行）。

③ 一方，“レッド・カー”の方では，このDOTの0か1で**ドット　の有る，無し**を判定します。

DOT（C3，C4）

《第3－29図》マイカーとレッド・カー

《第3－30図》右上コーナーでは

の値（0か1）により

TRACE$（0）＝“　”：ドット・なし

TRACE$（1）＝“・”：ドット・あり

をPRINTします（1710行）。TRACE$については，176行で定義してあります。

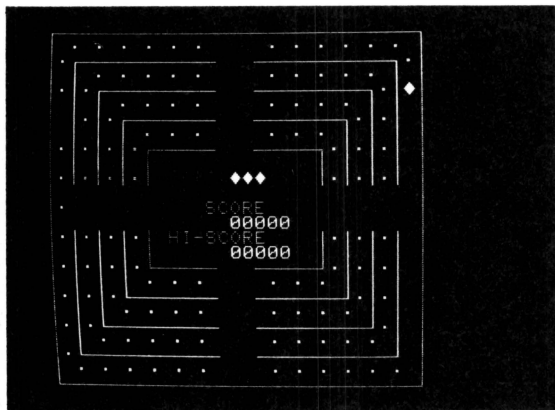
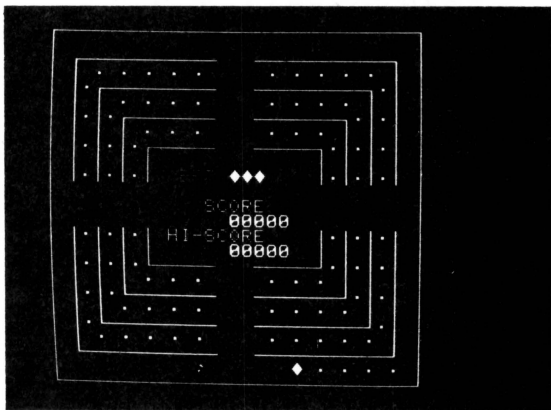
これでドット・の問題がすべて解決しました。どっと疲れましたか？　フー！

おわりに

今章は内容がハードであったため，さぞかしお疲れになったこととお察しします。たとえば最初にいきなり構造化プログラミングの核心の部分を紹介したり，ハア，まったく御苦労様でした。

しかし，しかし。自分で組んだプログラムがうまく動くと，それまでの苦心が一ぺんに吹き飛んでしまいます。さあ，ここでリストを走らせてみましょう。写真9，写真10を見てください。とくに“レッド・カー”の軌跡に注意して。写真9では，ドット・が残っていますが，写真10ではすでに空白の上を走っていますからちゃんとドット・が消えていますね？

あなたも自分で作ったプログラムを走らせてみてください。二つの車が元気よく，クルクル走っていますか？　気分爽快でしょう？　飽きるまで見ていてください。きっと目がまわりますよ。

《第3－31図》プログラミング哀歌

《写真9》レッド・カーのドットが残っている（交差前）

《写真10》レッド・カーのドットが残っていない（交差後）

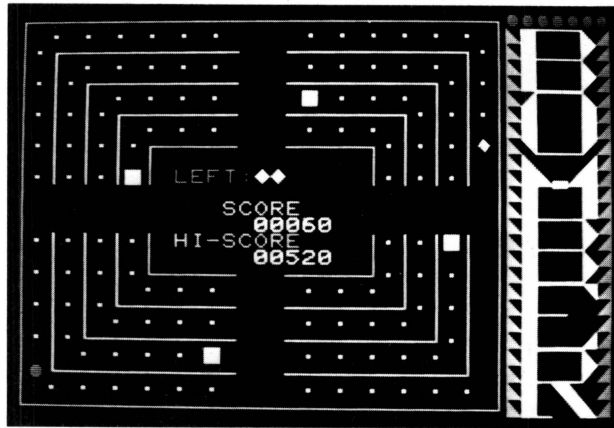
《リスト3－14》BOMBERプログラムリスト(第3章)

```
10 '♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
12 '♥ BOMBER for 'INVITATION FOR GAMING'2-3 ♥
14 '♥              << 82.2.15-X.XX >>       ♥
16 '♥                        by K.TUKAGOSHI ♥
18 '♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
19 '
20 '—— VARIABLE ——
22 'HISCR             :ﾊｲ ｽｺｱ
24 'LEFT              :COUNTER OF LEFT CAR
26 'SCR               :ｽｺｱ
28 'XMYCAR,YMYCAR     :LOCATE MYCAR
30 'C1,C2             :MYCAR ｺｰｽ & ｼﾝﾛ
32 'MCRS              :DIRECTION OF MYCAR      (1=ﾐｷﾞ,2=ｳｴ,3=ﾋﾀﾞﾘ,4=ｼﾀ)
34 'CMEMO             :MEMO of MCRS(C1,C2) OR RCRS(C3,C4)
36 'XRED,YRED         :LOCATE REDCAR
38 'C3,C4             :REDCAR ｺｰｽ & ｼﾝﾛ
40 'RCRS              :DIRECTION OF REDCAR
58 '
59 '—— DIMENSION ——
60 'CRS(ｺｰｽ,ｷｮﾘ)      :ﾜｸ =0,1,2,3 (0=OUTSIDE,3=INSIDE)
61 '                  :ｼﾝﾛ=0     ｾﾞﾝｼﾝ  [ﾂｷﾞ ﾊ ﾏｲｶｰ ﾉ ﾊﾞｱｲ]
62 '                  :    =1-4  ﾎｳｺｳ ﾃﾝｶﾝ       (1=ﾐｷﾞ,2=ｳｴ,3=ﾋﾀﾞﾘ,4=ｼﾀ)
63 '                  :    =5-8  ｷｰｽｷｬﾝ 1 ﾎｳｺｳ (5=ﾐｷﾞ,6=ｳｴ,7=ﾋﾀﾞﾘ,8=ｼﾀ)
64 '                  :    =9,10 ｷｰｽｷｬﾝ 2 ﾎｳｺｳ (9=ｻﾕｳ,10=ｼﾞｮｳｹﾞ)
65 '                  :    =99   ｼﾝﾛ ｶｳﾝﾀｰ=0 ﾍ
66 'XADD(X),YADD(Y)   :ﾍﾝｲ X,Y
68 'DOT(ﾜｸ,ｷｮﾘ)       :0=" ",1="・"
70 'TRACE$(1)         :0=" ",1="・"
99 '
100 '▬▬▬ MAIN ROUTINE ▬▬▬
110 '—— COLD START ——
115 DEFINT A-Z                                'SET INTEGER
120 HISCR=0
150 DIM CRS(3,49),ADD(4,4),DOT(3,49)
160   RESTORE 9110                            'READ CRS(3,50)
162   FOR I=0 TO 3
164     FOR J=0 TO 49
166       READ CRS(I,J)
168   NEXT J,I
170   FOR I=1 TO 4                            'READ XADD(4),YADD(4)
172     READ XADD(I),YADD(I)
174   NEXT
176   TRACE$(0)=" ":TRACE$(1)="・"             'SET TRACE$
199 '
200 '—— HOT  START ——
205 FOR I=0 TO 3:FOR J=0 TO 49:DOT(I,J)=1:NEXT J,I   'SET DOT
210 SCR=0:LEFT=3
220 XMYCAR=29:YMYCAR=22:C1=0:C2=0 :MCRS=2     'RESET MYCAR
230 XRED=1   :YRED=22  :C3=0:C4=35:RCRS=2     'RESET REDCAR
250 WIDTH 40,25:CONSOLE 0,25,0,1:COLOR 5,32,0:PRINT CHR$(12);
260 GOSUB 1000                                'ｶﾞﾒﾝ
270   COLOR 7:LOCATE XMYCAR,YMYCAR:PRINT "◆";
275   COLOR 2:LOCATE XRED,YRED:PRINT "●";
276 '
277 '—— GAME ——
280 GOSUB 1600                               'MOVE MYCAR
290 GOSUB 1700                               'MOVE REDCAR
900 GOTO 280                                 'MAIN END
999 '
1000 '▬▬▬ SUB ROUTINE ▬▬▬
1001 '—— ｶﾞﾒﾝ ——
1010 COLOR 5:PRINT CHR$(12);                 'PRINT ・
1020 FOR Y=1 TO 23
1022   X=(Y MOD 2)+1
1024   FOR X=X TO X+28 STEP 2
1026     LOCATE X,Y:PRINT "・";
1028 NEXT X,Y
1030 FOR I=0 TO 8 STEP 2                     'PRINT RECTANGLE
1031   IF (I MOD 8)=0 THEN COLOR 1 ELSE COLOR 5
1032   LOCATE I,I:PRINT "┌";
1033   Y1=I:Y2=24-I
1034   FOR X=I+1 TO 29-I
1036     LOCATE X,Y1:PRINT "─";:LOCATE X,Y2:PRINT "─";
1038   NEXT
1039   LOCATE X,Y1:PRINT "┐";:LOCATE X,Y2:PRINT "┘";
1040   X1=I:X2=30-I
1042   FOR Y=I+1 TO 23-I
1044     LOCATE X1,Y:PRINT "│";:LOCATE X2,Y:PRINT "│";
```

```
1046    NEXT
1048    LOCATE X1,Y:PRINT "└";
1050 NEXT
1060 X1=9 :X2=21:Y1=9 :Y2=15:GOSUB 1100          'CROSS OUT
1062 X1=14:X2=16:Y1=1 :Y2=23:GOSUB 1100
1064 X1=1 :X2=29:Y1=11:Y2=13:GOSUB 1100
1070 RESTORE 9020                                'PRINT TITLE
1071 FOR Y=1 TO 24
1072   READ I$:COLOR 2:LOCATE 31,Y:PRINT "◣";:COLOR 6:PRINT I$;
1073   COLOR 2:PRINT "◢";
1074 NEXT
1080 COLOR 2:LOCATE 10,10:PRINT "LEFT:    ";:GOSUB 1200 'PRINT MESSAGE
1082 COLOR 4:LOCATE 13,12:PRINT    "SCORE";:GOSUB 1300
1084 COLOR 4:LOCATE 10,14:PRINT "HI-SCORE";:GOSUB 1400
1090 RETURN
1099 '
1100 '—— CLEAR RECTANGLE ——————
1101 ' ——— PARA IN:X1,X2,Y1,Y2 ——
1110 FOR Y=Y1 TO Y2
1120   FOR X=X1 TO X2
1130     LOCATE X,Y:PRINT " ";
1140 NEXT X,Y:RETURN
1199 '
1200 '—— PRINT LEFT ——
1210 IF LEFT=0 THEN RETURN                       'LEFT=0 THEN ﾋｮｳｼﾞ ｾｽﾞ
1220 FOR X=15 TO 14+LEFT
1230   COLOR 7:LOCATE X,10:PRINT "◆";
1240 NEXT:RETURN
1299 '
1300 '—— PRINT SCORE ——
1310 COLOR 7:LOCATE 15,13:PRINT RIGHT$("0000"+HEX$(SCR),5):RETURN
1399 '
1400 '—— PRINT HI-SCORE ——
1410 COLOR 7:LOCATE 15,15:PRINT RIGHT$("0000"+HEX$(HISCR),5):RETURN
1499 '
1500 '—— CHANGE DIRECTION MYCAR ——
1510 XMYCAR=XMYCAR-1:YMYCAR=YMYCAR+1:MCRS=1:RETURN     'FOR RIGHT
1520 XMYCAR=XMYCAR+1:YMYCAR=YMYCAR+1:MCRS=2:RETURN     'FOR UP
1530 XMYCAR=XMYCAR+1:YMYCAR=YMYCAR-1:MCRS=3:RETURN     'FOR LEFT
1540 XMYCAR=XMYCAR-1:YMYCAR=YMYCAR-1:MCRS=4:RETURN     'FOR DOWN
1599 '
1600 '—— MOVE MYCAR ——
1610 COLOR 5:LOCATE XMYCAR,YMYCAR:PRINT " ";    'ERASE MYCAR
1620   CMEMO=CRS(C1,C2)                          'CMEMO=ｼﾝﾛ
1622   DOT(C1,C2)=0                              'ERASE DOT
1625   IF CMEMO=99 THEN C2=C2+1:GOTO 1620        'SKIP 99
1630   C2=C2+1:IF C2=50 THEN C2=0                '1-ROUND END ?
1640   ON CMEMO GOSUB 1510,1520,1530,1540,2100,2200,2300,2400,2500,2600
1650   XMYCAR=XMYCAR+XADD(MCRS):YMYCAR=YMYCAR+YADD(MCRS)
1660 COLOR 7:LOCATE XMYCAR,YMYCAR:PRINT "◆";    'PRINT MYCAR
1690 RETURN
1699 '
1700 '—— MOVE REDCAR ——
1710 COLOR 5:LOCATE XRED,YRED:PRINT TRACE$(DOT(C3,C4)); 'ERASE REDCAR
1715   C4=C4-1:IF C4<0 THEN C4=49                '1-ROUND END ?
1720   CMEMO=CRS(C3,C4)                          'CMEMO=ｼﾝﾛ
1730   IF CMEMO=99 THEN C4=C4-1:GOTO 1720        'SKIP 99
1740   ON CMEMO GOSUB 1820,1810,1840,1830,2700,2800,2900,3000,3100,3200
1750   XRED=XRED+XADD(RCRS):YRED=YRED+YADD(RCRS)
1760 COLOR 2:LOCATE XRED,YRED:PRINT "●";        'PRINT REDCAR
1790 RETURN
1799 '
1800 '—— CHANGE DIRECTION REDCAR ——
1810 XRED=XRED+1:YRED=YRED+1:RCRS=3:RETURN       'FOR LEFT
1820 XRED=XRED-1:YRED=YRED+1:RCRS=2:RETURN       'FOR UP
1830 XRED=XRED-1:YRED=YRED-1:RCRS=1:RETURN       'FOR RIGHT
1840 XRED=XRED+1:YRED=YRED-1:RCRS=4:RETURN       'FOR DOWN
1850 '
2100 '—— GO RIGHT? MYCAR ——
2110 CMEMO=0
2190 RETURN
2200 '—— GO UP? MYCAR ——
2210 CMEMO=0
2290 RETURN
2300 '—— GO LEFT? MYCAR ——
2310 CMEMO=0
2390 RETURN
2400 '—— GO DOWN? MYCAR ——
```

```
2410 CMEMO=0
2490 RETURN
2500 '—— GO LEFT OR RIGHT? MYCAR ——
2510 CMEMO=0
2590 RETURN
2600 '—— GO UP OR DOWN? MYCAR ——
2610 CMEMO=0
2690 RETURN
2700 '—— GO RIGHT? MYCAR ——
2710 CMEMO=0
2790 RETURN
2800 '—— GO UP? MYCAR ——
2810 CMEMO=0
2890 RETURN
2900 '—— GO LEFT? MYCAR ——
2910 CMEMO=0
2990 RETURN
3000 '—— GO DOWN? MYCAR ——
3010 CMEMO=0
3090 RETURN
3100 '—— GO LEFT OR RIGHT? MYCAR ——
3110 CMEMO=0
3190 RETURN
3200 '—— GO UP OR DOWN? MYCAR ——
3210 CMEMO=0
3290 RETURN
8999 '
9000 '▬▬▬ DATA AREA ▬▬▬
9010 '—— TITLE DATA ——
9020  DATA "█▔▔▔◣"
9022  DATA "█▁▁▁◤"
9024  DATA "█   ◣"
9026  DATA "█▁▁▁◤"
9028  DATA "◢▔▔▔◣"
9030  DATA "█   █"
9032  DATA "█   █"
9034  DATA "◥▁▁▁◤"
9036  DATA "◣   ◢"
9038  DATA "█◣▂◢█"
9040  DATA "█ ▔ █"
9042  DATA "█   █"
9044  DATA "█▔▔▔◣"
9046  DATA "█▁▁▁◤"
9048  DATA "█   ◣"
9050  DATA "█▁▁▁◤"
9052  DATA "█▔▔▔◥"
9054  DATA "█▁▁  "
9056  DATA "█    "
9058  DATA "█▁▁▁◢"
9060  DATA "█▔▔▔◣"
9062  DATA "█▁▁▁◤"
9064  DATA "█ ◥◣ "
9066  DATA "█▁ ◥◣"
9098 '
9099 '—— COURSE DATA ——
9110 DATA 00,00,00,00,07,07,00,00,00,00,03:              'COURSE-0
9111 DATA 00,00,00,00,00,08,08,08,00,00,00,00,00,04
9112 DATA 00,00,00,00,05,05,00,00,00,00,01
9114 DATA 00,00,00,00,00,06,06,06,00,00,00,00,00,02
9116 DATA 00,00,00,99,09,09,00,00,00,99,03:              'COURSE-1
9117 DATA 00,00,00,00,99,10,10,10,00,00,00,00,99,04
9118 DATA 00,00,00,99,09,09,00,00,00,99,01
9120 DATA 00,00,00,00,99,10,10,10,00,00,00,00,99,02
9122 DATA 00,00,99,99,09,09,00,00,99,99,03:              'COURSE-2
9123 DATA 00,00,00,99,99,10,10,10,00,00,00,99,99,04
9124 DATA 00,00,99,99,09,09,00,00,99,99,01
9126 DATA 00,00,00,99,99,10,10,10,00,00,00,99,99,02
9128 DATA 00,99,99,99,05,05,00,99,99,99,03:              'COURSE-3
9129 DATA 00,00,99,99,99,06,06,06,00,00,99,99,99,04
9130 DATA 00,99,99,99,07,07,00,99,99,99,01
9132 DATA 00,00,99,99,99,08,08,08,00,00,99,99,99,02
9139 '
9200 '—— ADD DATA ——
9210 DATA +2, 0:                    'RIGHT
9212 DATA  0,-2:                    'UP
9214 DATA -2, 0:                    'LEFT
9216 DATA  0,+2:                    'DOWN
```

リアルタイムゲーム「BOMBER」に挑戦

第4章 キー入力について

“春過ぎて
夏きにけらし
我がソフト
走る姿よ
天のバク山”
(珍古今集　作品2番)

はじめに

今年も‘**マイコンショウ**’が終り，また新しい機種が発表・発売されようとしています。平和島の東京会場では昨年の倍のスペースをさき，それでもかなりの混雑でした(**第3－32図**)。

ソフトウェア・ハウスや周辺機器メーカーの台頭と同時にマシンの高性能化，低価格化，および操作性の高上に伴い，一般サラリーマンに加えて主婦・OLといった従来マイコン・ホビイストの周辺に位置していた**潜在的需要層**が，今後新しいマイコン・ユーザーとして業界へ流入しようとしています。それはショウの中でもハッキリ目立ち，またとりわけホビー関係のブースでの混雑を見ても明らかでした。これらの動きは当然歓迎されるべきで，彼等の中から**新感覚による新しいGAME**が生まれることが期待されます。

思えば1972年(昭和47年)，国産マイコンの第1号が九州日本電気より商品化され，1976年(昭和51年)頃から始まった**マイコン・ホビー**も隔世の感があります。たとえば1979年(昭和54年)1月1日付「朝日新聞」で都内在住356人を対象に

“どんな21世紀を思い描くか？”

というテーマでアンケート調査の結果をまとめています。この中で，

“新・三種の神器”

と題して，

『実際，62%の人は，主婦がコンピュータのボタンを押す時代を予想しており，家庭の「新・三種の神器」のひとつにコンピュータが加えられるのもそう遠くはないともとれる。ある予備校生は，「21世紀になって

《第3－32図》 マイコンショウ’83

も，四畳半の下宿族は依然存在し続け，コタツの上に置いたコンピュータの端末をポソリと押しては，アルバイトの口をさがす，てな調子じゃないかな」といっていた』
と報告されています。あなたは，この約4年前のアンケート結果を御覧になってどのように思われたでしょうか？ 重要なことは，この当時ですら“マイコン”，と言っても世間にはまったく知られていなかったということです。信じられますか？
マア何はともあれ，BOMBER作りを楽しんでまいりましょう。バビューン！

第4章の目標

第3章の

SKIPマーク

理解できましたか？
——良くわからん。
——説明が悪い！
ごもっともです。スイマセン
一応第3章までで各車の動き

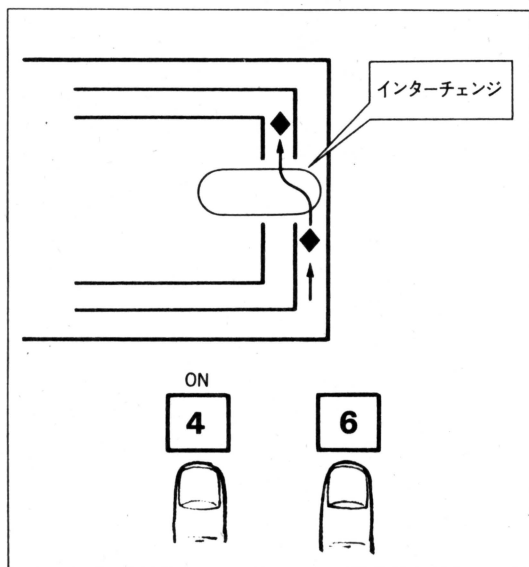
MYCAR——左まわりに
REDCAR——右まわりに

が成功しています。そこでこの章ではさらに一歩進め，

キー入力

をキャッチし，その**キーの種類**に応じて

MYCARのコースを変える

ことに挑戦してみましょう。

《第3−33図》 コースを変える

ペコちゃん，ポコちゃんの指摘

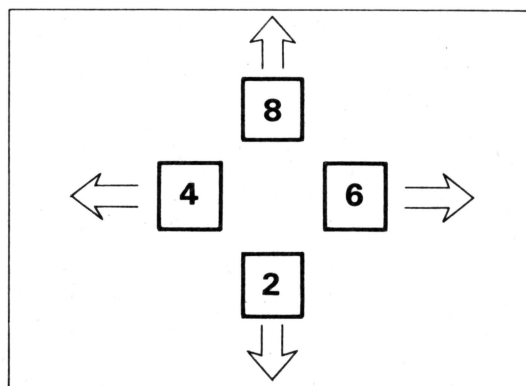
ポコちゃん：キーの種類に応じてMYCARのコースを変えるんだって。何，それ？
ペコちゃん：だからインターチェンジのところにMYCARが来たとき，もしキーが押されていたらコースを変えてあげるのよ（**第3−33図**）。
ポコちゃん：でもさ，キーの種類はマシンによって異なるよ。そうすると，マシンによってプログラムを変えなきゃいけないの？
ペコちゃん：そうね。**機械によってキーの配置が異なる**わね。“GAMINGへの招待”の今度のシリーズでは，マシンの違いを考慮すると言っているわね。どうするのかしら？

これから我々が挑戦しようとしていることは，ペコちゃん，ポコちゃん御指摘のように，**マシンの違い**によって異なってきます。それは，次の二点においてです。

- **キーの配置が異なる**
- **キー入力の命令が異なる**

たとえば私のPC版では，前に設定しましたように**第3−34図**のようになっていますが，これは当然マシンによって異なります。また，

リアルタイムキー入力

の命令もマシンによって異なります。以上の理由により，プログラムはマシンによって異なってしまうことは避けられません。
そこで——。

《第3−34図》 キー・ファンクション

移植可能なプログラム

かつて（先のアンケートの行なわれた約一年位前まで）マイコンの世界は，自作，キット等による**ワンボード**のものが全盛で，当然オリジナルでは**原始的なマシン語**しか走りませんでした。その間マニアによる多くの**言語プロセッサ**が発表され，各自，自分のマシンにそれらを移植しようと必死になっていました。

当時のマシンは，外国製品あり，国産製品あり，自作品ありでこれこそ千差万別でした。ところがそんなバラバラなマシンの中にあって，比較的多くの機種に移植されたいくつかの言語プロセッサがあります。そしてそれらには共通の設計方針がありました。

すなわちそれらの言語プロセッサは，最初から

移植されること

を意識して設計がなされており，次のような方針にしたがっていました。

普通ハードウエアによるマシンの差は，ソフトウエアの面から見れば，**周辺機器との入出力の方法**に絞られます。当時の平均的な周辺機器は，

入力装置：キーボード等
出力装置：ＣＲＴ，７セグメントＬＥＤ，
プリンタ等

でしたから，**これらの機器とのやりとりをサポートするルーチン**を各自に作ってもらい，その**入口となるアドレスは統一**してしまえば，異機種の移植は容易となります。

たとえば，基本的な入出力サポート・ルーチンは次のとおりです。

① ＧＥＴＣ (get character)
一文字入力ルーチン。キーの入力があれば，そのキャラクタ・コードをＡレジスタに入れ，かつ出力装置（たとえばＣＲＴ）にエコー・バックし，ＲＥＴする。

② ＰＵＴＣ (put character)
一文字出力ルーチン。Ａレジスタのキャラクタ・コードを出力装置に出力する。

③ ＩＮＫＥＹ
キー入力センス・ルーチン。キーボード等入力装置からの入力があったかをチェックする。もし入力があれば，

ＣＹ＝１（ＣＹ：キャリー・フラグ）

でＲＥＴする。

以上の三つの基本ルーチンだけは，各自のマシンに合わせて作ってもらい，言語プロセッサ本体の中では次のように配置して使います。

```
;
; ××   PROCESSOR
;
GETC:    JP  ××
PUTC:    JP  ××
INKEY:   JP  ××
;
MAIN:    ...............
      ∫
```

つまり，こういう書き方をすれば，〰〰の**わずか６バイト**分だけを各自のマシンに合わせるだけであとは

すべて同じプログラム

にすることができるわけです。あとはこのプログラムを各自のマシンの**適当なＲＡＭ領域にリロケート**すれば，無事**移植は完了**します。

基本操作 getc

ポコちゃん：何か急に難しくなったよ。
ペコちゃん：良くわからないわね。
——**ツカ**：エート，エート，今日の話しは良くわからなくても結構です。もう少し先まで我慢して読んでいただくとわかってきますよ。
もう一つ例をあげましょう。あなたは，

“ソフトウエア作法
Brian W. Kernighan
and P. J. Plauger
（共立出版）”

という本を御存知でしょうか？　この種の本にあまり馴染みのない人には少々取りつきにくい本かもしれませんが，あなたが**今後もソフトウエア作りを続けていく**予定があるようでしたら，この本を一読することをお勧めします。言語は

RATFOR
（ＲＡＴＩＯＮＡＬ　ＦＯＲＴＲＡＮ）

という構造化言語（最終的にはＦＯＲＴＲＡＮ自身で記述された変換プログラムにより，ＦＯＲＴＲＡＮに変換される）で書かれていますが，最初の文法から解説されていますので，ある程度プログラミングをやったことのある人なら十分理解できるでしょう。

この本は，全体が一つの哲学をもって記述されています。すなわち，

「分野のいかんを問わず，実質的な前進をとげるための唯一の方法は，他人の仕事を利用することである。にもかかわらず，プログラマたちはすでに存在しているプログラムを使おうとせず，応用ごとに新しくプログラムを作りたがる。」

「本書では可能な限り，複雑なプログラムを簡単なプログラムをもとにして組み立てるようにする。また可能ならば，すでに存在している道具を利用することにより，またはその組み合わせについて新しい使い方を見つけ出すことによって，**プログラムを作らずにすますよう**にする。」

「この本の終りまで読み進んだとき読者は，プログラマとして読者が出会う問題の多くを解決するソフトウエア的な道具群（注，5000行を越えるプログラム）を手に入れていることになろう。」

ということで，沢山の**有益なモジュール群**を

もっとも利用しやすい形

で作っていき，プログラム作りはそれら**既存のモジュール群を利用することで**

生産性を高めよう！

とするものです。

この書においてもこれらのモジュール群を作り上げるにあたり，最初に

マシンによる入出力操作の違い

という問題に当たっています。これをカモフラージュするため，この本では種々のくふうがなされています。たとえば，何らかの入力源から読み出す関数として

getc

というものを登場させています。そして，

『これが「外界」に対する窓口となる。オペレーティングシステム固有の入出力ルーチンを呼び出す仕事はそちらの方でやってくれる。……基本操作（ここでは **getc** のこと）を使えば，特定のオペレーティングシステムのくせに依存しすぎることのないプログラムを設計，製作することができる。』

と述べています。

さて，聡明なあなたは，この

getc

と先の言語プロセッサにおける

入出力基本ルーチン

は，考え方が似ていることに気がつかれたことでしょう。

《第３－35図》仮想マシン GAME 999のキー配置

仮想マシンGAME999

これから′ＢＯＭＢＥＲ′の製作にあたり，たった一つの

サブルーチン

を，

あなた自身の手で

あなたのマシンに合わせて

製作していただくことになります。その**仕様**については私が規定しますが，その**中身**についてはあなたにおまかせします。私は関知致しません。それは，今後のプログラムの進展に伴い，

マシンによる差

機種による差

を最少限にくい止めるためです。

それでは，どんなサブルーチンを作っていただくのか，その仕様を順に御説明致しましょう。それには，マシンによる不公平をなくすため，ここで仮想のパーソナルコンピュータ

ＧＡＭＥ９９９

を登場させ，そのマシンの上で説明していくことにします。

まずＭＹＣＡＲを動かすための，キーの配置を決めます。ＧＡＭＥ９９９では，**第３－35図**のようになりました。そこでサブルーチンの内容です。仮にこのサブルーチンの名前を

ＲＥＡＬ

と命名することにします。ＲＥＡＬは，メインルーチンからＣＡＬＬされると，すぐに

その時点で**キーボードが押されているか？**

チェックします。そしてその結果を

文字変数**ＫＹ＄**

に入れてＲＥＴＵＲＮします。その内容は次の通りです。

[マ]のキーが押されていた

（右へ動かそうとしていた）：ＫＹ＄＝″Ｒ″

［イ］のキーが押されていた
（上へ動かそうとしていた）：KY＄＝″U″
［コ］のキーが押されていた
（左へ動かそうとしていた）：KY＄＝″L″
［ン］のキーが押されていた
（下へ動かそうとしていた）：KYS＝″D″
そして**それ以外のキー（［マ］，［イ］，［コ］，［ン］以外のキー）が押されていた**か，または**キーが全然押されていなかった時**には，

KY＄＝″″
（ヌル・ストリング）

を入れてRETURNします。たとえば，サブルーチンREALをCALLすると，ほとんど瞬間的にメインルーチンにRETURNしてきます。そしてそのときのKY＄の値を調べ，仮に

KY＄＝″U″

であれば，ゲームをしている人はMYCARを

上に動かそうとしていた

と判定できますし，

KY＄＝″″

であれば，

MYCARを動かす意志はなかった

と判定できるわけです。

REALの実際

それでは，サブルーチンREALがプログラム上で実際どのように記述されるのか調べてみることにします。ここでも仮想マシン**GAME９９９**に登場願いましょう。

今仮にGAME９９９のリアルタイム・キー入力の関数が

GETKY$

であるとします。そしてその機能を次のように仮定します。

GETKY$
〈目的〉：リアルタイム・キー入力
〈書式〉：文字変数＝GETKY＄
〈機能〉：この関数が使用された時点でのキーの入力を調べる。**結果は文字変数**に代入される。その値は，
キーが押されていない＝″″
キーが押されていた　＝**キーの種類**

つまりキー［マ］が押されていれば，

文字変数＝″マ″

だし，何も押されていなければ，

文字変数＝″″

というわけです。

そこでこのGETKY＄を使ってサブルーチンREALを記述すると，次のようになります。

```
KY$=GETKY$ (入力文字をKY$へ)
IF  KY$="マ"
      THEN  KY$="R"
IF  KY$="イ"
      THEN  KY$="U"
IF  KY$="コ"
      THEN  KY$="L"
IF  KY$="ン"
      THEN  KY$="D"
IF  KY$ < > "R"  AND
    KY$ < > "U"  AND
    KYS < > "L"  AND
    KY$ < > "D"
            THEN  KY$=""
RETURN
```

以上が仮想マシンGAME９９９上におけるREALの見本です。もしこのサブルーチンをPC－8001で組むとしたら，リストの

3050行 ～ 3120行

のようになるでしょう。そしてこの

REAL

だけを各自が作っていただければ，その他の部分では

マシンによる**キー配置**の違い
マシンによる**キー入力命令**の違い

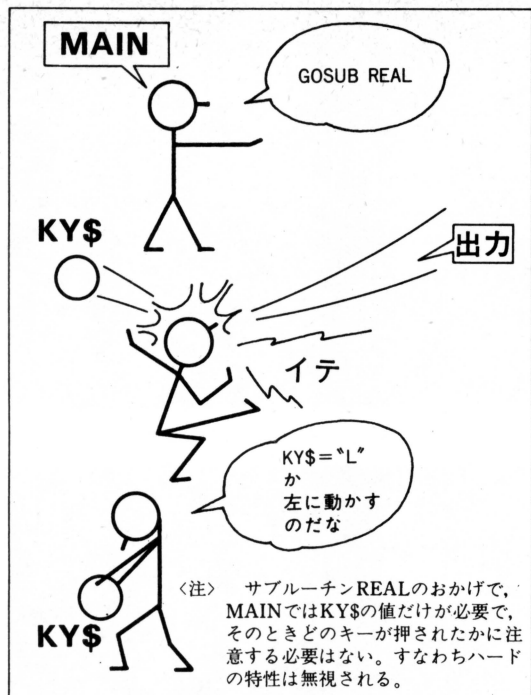
を**無視**することができるようになります。**KY＄の値だけに注目**すればよいのです（**第３－36図**）。

ここらあたりの事情，理解していただけたでしょうか，ポコちゃん，ペコちゃん？

コースを変更するために

リアルタイム・キー入力サブルーチンの規格品ができました。これさえできてしまえば，あとは話しが簡単です。

ペコちゃん：MYCARの動かし方，少しわかってきたわね。

ポコちゃん：ハテ？

《第3－36図》サブルーチンREAL

ペコちゃん：まずMYCARが第1インターチェンジまで着くと，**左のコース**に進路を変えられるわね。

ポコちゃん：それはわかる（**第3－37図**）。

ペコちゃん：そのときプログラムの中では，2640行からのサブルーチンに飛び込んでくるわけよ。

ポコちゃん：あっ，少しわかったぞ。するとそのサブルーチンをいじくってやればいいんだ！

ペコちゃん：そうよ。第3章までのリストだと，単に

CMEMO＝0

でRETURNしていただけたわ。

ポコちゃん：何を変えればいいのかな？

ペコちゃん：そこが問題よ。

ポコちゃん：悩める青春〜〜〜ムニャ，ムニャ。

進路変更のための二つの処理

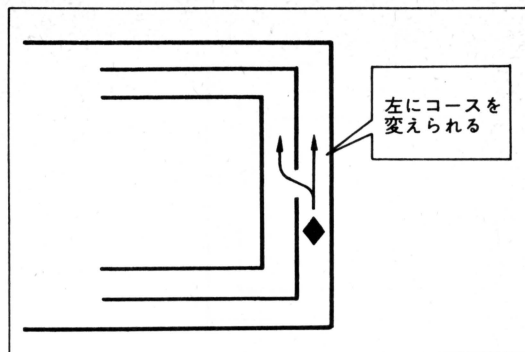
コースの変更について，ポコちゃん，ペコちゃんが良い指摘をしてくれました。そこで，この2640行のサブルーチンで何をすれば良いのか考えてみましょう。

まずゲームをおこなっている人が，

進路を左に取る意志があるのか？

確認します。つまり，

GOSUB REAL
↑
ここでは，3060

です。そして

《第3－37図》第1インターチェンジで

KY$＝″L″

かをチェックします。もしそうでなければ，ゲームをしている人は**進路変更の意志をもっていません**から，単純にRETURNします(2650行)。

さて,「**左への進路変更への意志あり！**」の場合です。このときは，次の**二点の作業**をおこないます。

① **MYCARの画面上の位置を左にする**

これは簡単です。

MYCARのX座標＝XMYCAR

ですから，

XMYCAR＝XMYCAR－2

でOKです。

② **C1の値を変更する。**

C1は，**MYCARのコース番号**を表わす変数でしたね。

C1＝0：1番外側のコース

〜

C1＝3：1番中側のコース

の**0〜3**の値を取り，**数字が大きくなる程中側のコース**を表わすわけです。進路変更するわけですから，当然C1は変更しなければなりません。

このC1の変更は，**2通りのケース**が考えられます。

1) **MYCARが上に向かうとき**

第3－38図①のケースです。このときは，**内側のコースへ進路変更**することになりますから，

C1＝C1＋1

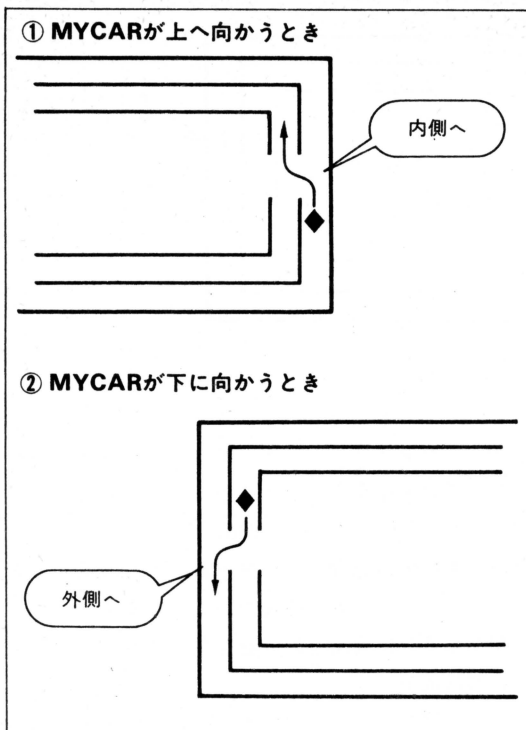
とします。

2) **MYCARが下に向かうとき**

第3－38図②のケースです。このときは，**外側のコース**へ進路変更することになりますから，

C1＝C1－1

とします。

現在MYCARがどちらに向かっているかは，

《第3－38図》左にコースを取るにも2ケースがある

MCRS

の値を調べればわかりますね？　これで2670行が何をしているのかおわかりになったと思います。

以上，ＭＹＣＡＲの進路変更を**左に変更する場合**について見てきました。他のケースについてもまったく同様にできますから，御自分で考えてみてくださいね。

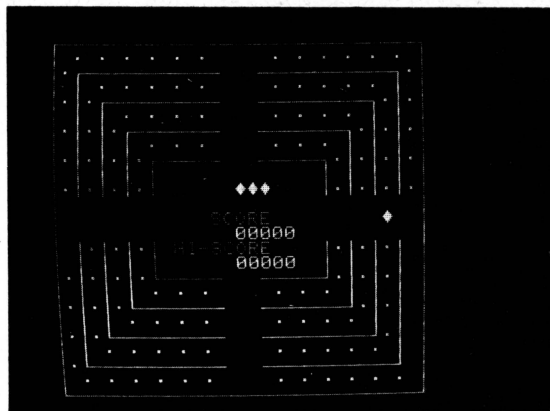
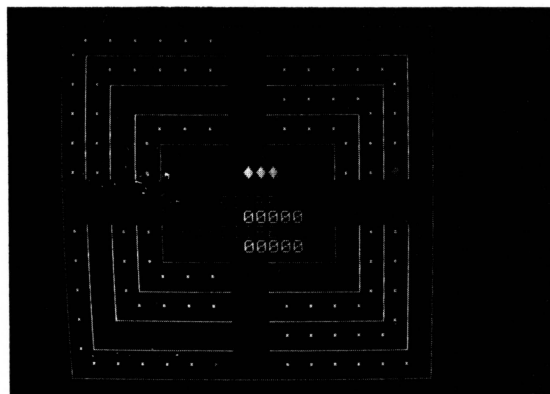
第4章のリストは，どの向きにも進路変更ができるようになっています。さっそく走らせてみましょう。**写真11**は，さっそく左側に進路をとったところです。ドット・のあとに御注目ください。また**写真12**は，あちこちに進路変更をしたのがわかります。**写真13**では，すべてのドット・を消しています。しかし，相変わらず**REDCARは外側のコースを回り続けています。**おろかですね？

インターチェンジのドット？

第3章のプログラムを走らせてみて，レッド・カーが最初のインターチェンジを通過後，**インターチェンジの真中にドット・を残して行く**――ということに気がつかれたでしょうか？

この原因は，最初に

DOT（3，49）

の設定のとき，

《写真11》　左にコースを変える

《写真12》　何度も進路変更

すべて1にしてある

からです。そのためレッド・カーはインターチェンジを通過後，そこにドット・があったものと勘違いしてしまうのです。

そこでＤＯＴの初期値をＤＡＴＡ文で用意する必要があります。3660行～3810行のＤＡＴＡがそれです。

また**リスト3－15**のリストでは，読み易くするため，

ＲＥＮＵＭ　１０００

を実行してあります。

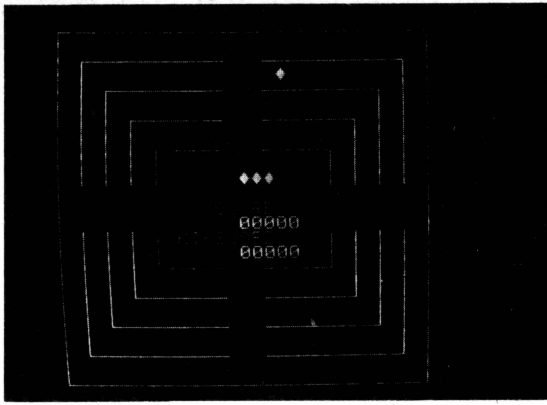
おわりに

“ＧＡＭＩＮＧへの招待”では，各章ごとのリストを載せています。しかし，これはあくまでも

私の**参考出品**

であり，実際のプログラムを作るのは，

あなた自身

です。実際にこのプログラムは，ＰＣ－8001の上で開発されていますから，たまたまあなたのマシンがＰＣであればリストをそのまま打ち込んでいただくだけで走るでしょう。しかし他機種の方でしたら，おそらく

《写真13》 すべてのドットを消す

一発では走らないことと思われます。重ねて言いますが，このリストはあくまでも

参考出品

です。

“GAMINGへの招待”について読者より御質問を載くことがあります。しかし中には，

「リストを打ち込んでみましたが走りません。どうしてでしょうか？　私の機種は××です」

の類のお手紙があったりして，非常に残念な思いがしたりすることがあります。“**GAMINGへの招待**”は，

GAMEのリストを届ける

のが目的ではありません。むしろそれ以前の

手作りによるGAMING

を狙っています。どうかその主旨を御理解戴き，あなたも**GAME作りの醍醐味**を味わって戴きたいと思います。

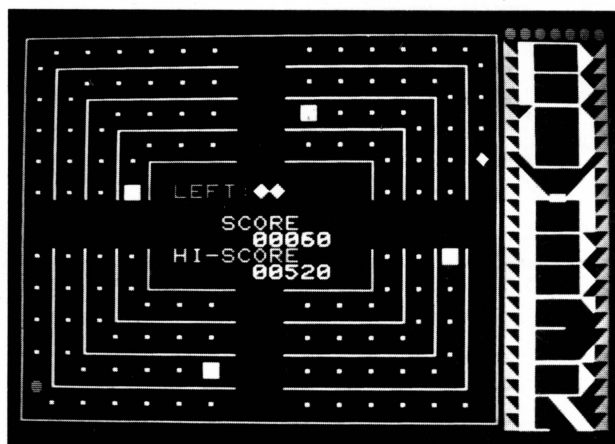
《リスト3－15》 BOMBERプログラムリスト(第4章)

```
1000 '♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
1010 '♥  BOMBER for 'INVITATION FOR GAMING'2-4 ♥
1020 '♥                 << 82.2.15-X.XX >>      ♥
1030 '♥                         by K.TUKAGOSHI ♥
1040 '♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
1050 '
1060 '—— VARIABLE ——
1070 'HISCR              :ハイ スコア
1080 'LEFT               :COUNTER OF LEFT CAR
1090 'SCR                :スコア
1100 'XMYCAR,YMYCAR      :LOCATE MYCAR
1110 'C1,C2              :MYCAR コース & キジュンテン カラ ノ キョリ
1120 'MCRS               :DIRECTION OF MYCAR     (1=ミギ,2=ウエ,3=ヒダリ,4=シタ)
1130 'CMEMO              :MEMO of MCRS(C1,C2) OR RCRS(C3,C4)
1140 'XRED,YRED          :LOCATE REDCAR
1150 'C3,C4              :REDCAR コース & キジュンテン カラ ノ キョリ
1160 'RCRS               :DIRECTION OF REDCAR
1170 'I0,I1              :VALUE OF INP
1180 'KY$                :VALUE OF KEYSCAN  (KEY OFF="")
1190 '
1200 '—— DIMENSION ——
1210 'CRS(コース,キョリ)    :ワク =0,1,2,3 (0=OUTSIDE,3=INSIDE)
1220 '                   :シンロ=0     ゼンシン  [ツギ ハ マイカー ノ バアイ]
1230 '                   :    =1-4  ホウコウ テンカン     (1=ミギ,2=ウエ,3=ヒダリ,4=シタ)
1240 '                   :    =5-8  キースキャン 1 ホウコウ (5=ミギ,6=ウエ,7=ヒダリ,8=シタ)
1250 '                   :    =9,10 キースキャン 2 ホウコウ (9=サユウ,10=ジョウゲ)
1260 '                   :    =99   シンロ カウンター=0 ヘ
1270 '                   :                  [ツギ ハ RED ノ バアイ]
1280 '                   :    =1-4  ホウコウ テンカン     (1=ウエ,2=ヒダリ,3=シタ,4=ミギ)
```

```
1290 'XADD(X),YADD(Y)  :ﾍﾝｲ X,Y
1300 'DOT(ﾜｸ,ｷｮﾘ)      :0=" ",1="・"
1310 'TRACE$(1)        :0=" ",1="・"
1320 '
1330 '▬▬▬ MAIN ROUTINE ▬▬▬
1340 '—— COLD START ——
1350 DEFINT A-Z'                                   SET INTEGER
1360 HISCR=0
1370 DIM CRS(3,49),ADD(4,4),DOT(3,49)
1380   RESTORE 3430'                               READ CRS(3,49)
1390   FOR I=0 TO 3
1400     FOR J=0 TO 49
1410       READ CRS(I,J)
1420   NEXT J,I
1430   FOR I=1 TO 4'                               READ XADD(4),YADD(4)
1440     READ XADD(I),YADD(I)
1450   NEXT
1460   TRACE$(0)=" ":TRACE$(1)="・"'               SET TRACE$
1470 '
1480 '—— HOT  START ——
1490 RESTORE 3660'                                 SET DOT
1500 FOR I=0 TO 3
1510   FOR J=0 TO 49
1520     READ DOT(I,J)
1530 NEXT J,I
1540 SCR=0:LEFT=3
1550 XMYCAR=29:YMYCAR=22:C1=0:C2=0 :MCRS=2'        RESET MYCAR
1560 XRED=1   :YRED=22  :C3=0:C4=35:RCRS=2'        RESET REDCAR
1570 WIDTH 40,25:CONSOLE 0,25,0,1:COLOR 5,32,0:PRINT CHR$(12);
1580 GOSUB 1670'                                   ｶﾞﾒﾝ
1590   COLOR 7:LOCATE XMYCAR,YMYCAR:PRINT "◆";
1600   COLOR 2:LOCATE XRED,YRED:PRINT "●";
1610 '
1620 '—— GAME ——
1630 GOSUB 2270'                                   MOVE MYCAR
1640 GOSUB 2380'                                   MOVE REDCAR
1650 GOTO 1630'                                    MAIN END
1660 '
1670 '▬▬▬ SUB ROUTINE ▬▬▬
1680 '—— ｶﾞﾒﾝ ——
1690 COLOR 5:PRINT CHR$(12);'                      PRINT ・
1700 FOR Y=1 TO 23
1710   X=(Y MOD 2)+1
1720   FOR X=X TO X+28 STEP 2
1730     LOCATE X,Y:PRINT "・";
1740 NEXT X,Y
1750 FOR I=0 TO 8 STEP 2'                          PRINT RECTANGLE
1760   IF (I MOD 8)=0 THEN COLOR 1 ELSE COLOR 5
1770   LOCATE I,I:PRINT "┌";
1780   Y1=I:Y2=24-I
1790   FOR X=I+1 TO 29-I
1800     LOCATE X,Y1:PRINT "─";:LOCATE X,Y2:PRINT "─";
1810   NEXT
1820   LOCATE X,Y1:PRINT "┐";:LOCATE X,Y2:PRINT "┘";
1830   X1=I:X2=30-I
1840   FOR Y=I+1 TO 23-I
1850     LOCATE X1,Y:PRINT "|";:LOCATE X2,Y:PRINT "|";
1860   NEXT
1870   LOCATE X1,Y:PRINT "└";
1880 NEXT
1890 X1=9 :X2=21:Y1=9 :Y2=15:GOSUB 2020'         CROSS OUT
1900 X1=14:X2=16:Y1=1 :Y2=23:GOSUB 2020
1910 X1=1 :X2=29:Y1=11:Y2=13:GOSUB 2020
1920 RESTORE 3160'                               PRINT TITLE
1930 FOR Y=1 TO 24
1940   READ I$:COLOR 2:LOCATE 31,Y:PRINT "◣";:COLOR 6:PRINT I$;
1950   COLOR 2:PRINT "◢";
1960 NEXT
1970 COLOR 2:LOCATE 10,10:PRINT "LEFT:   ";:GOSUB 2090 'PRINT MESSAGE
1980 COLOR 4:LOCATE 13,12:PRINT    "SCORE";:GOSUB 2150
1990 COLOR 4:LOCATE 10,14:PRINT "HI-SCORE";:GOSUB 2180
2000 RETURN
2010 '
2020 '—— CLEAR RECTANGLE ————
2030 ' —— PARA IN:X1,X2,Y1,Y2 —
2040 FOR Y=Y1 TO Y2
2050   FOR X=X1 TO X2
2060     LOCATE X,Y:PRINT " ";
2070 NEXT X,Y:RETURN
2080 '
2090 '—— PRINT LEFT ——
2100 IF LEFT=0 THEN RETURN'                      LEFT=0 THEN ﾋｮｳｼﾞ ｾｽﾞ
2110 FOR X=15 TO 14+LEFT
2120   COLOR 7:LOCATE X.10:PRINT "◆";
```

```
2130 NEXT:RETURN
2140 '
2150 '—— PRINT SCORE ——
2160 COLOR 7:LOCATE 15,13:PRINT RIGHT$("0000"+HEX$(SCR),5):RETURN
2170 '
2180 '—— PRINT HI-SCORE ——
2190 COLOR 7:LOCATE 15,15:PRINT RIGHT$("0000"+HEX$(HISCR),5):RETURN
2200 '
2210 '—— CHANGE DIRECTION MYCAR ——
2220 XMYCAR=XMYCAR-1:YMYCAR=YMYCAR+1:MCRS=1:RETURN'    FOR RIGHT
2230 XMYCAR=XMYCAR+1:YMYCAR=YMYCAR+1:MCRS=2:RETURN'    FOR UP
2240 XMYCAR=XMYCAR+1:YMYCAR=YMYCAR-1:MCRS=3:RETURN'    FOR LEFT
2250 XMYCAR=XMYCAR-1:YMYCAR=YMYCAR-1:MCRS=4:RETURN'    FOR DOWN
2260 '
2270 '—— MOVE MYCAR ——
2280 COLOR 5:LOCATE XMYCAR,YMYCAR:PRINT " ";'     ERASE MYCAR
2290   CMEMO=CRS(C1,C2)'                            CMEMO=ｼﾝﾛ
2300   DOT(C1,C2)=0'                                ERASE DOT
2310   IF CMEMO=99 THEN C2=C2+1:GOTO 2290'          SKIP 99
2320   C2=C2+1:IF C2=50 THEN C2=0'                  1-ROUND END ?
2330   ON CMEMO GOSUB 2220,2230,2240,2250,2550,2600,2650,2700,2750,2800
2340   XMYCAR=XMYCAR+XADD(MCRS):YMYCAR=YMYCAR+YADD(MCRS)
2350 COLOR 7:LOCATE XMYCAR,YMYCAR:PRINT "◆";'     PRINT MYCAR
2360 RETURN
2370 '
2380 '—— MOVE REDCAR ——
2390 COLOR 5:LOCATE XRED,YRED:PRINT TRACE$(DOT(C3,C4));'   ERASE REDCAR
2400   C4=C4-1:IF C4<0 THEN C4=49'                  1-ROUND END ?
2410   CMEMO=CRS(C3,C4)'                            CMEMO=ｼﾝﾛ
2420   IF CMEMO=99 THEN C4=C4-1:GOTO 2410'          SKIP 99
2430   ON CMEMO GOSUB 2500,2490,2520,2510,2850,2900,2930,2960,2990,3020
2440   XRED=XRED+XADD(RCRS):YRED=YRED+YADD(RCRS)
2450 COLOR 2:LOCATE XRED,YRED:PRINT "●";'         PRINT REDCAR
2460 RETURN
2470 '
2480 '—— CHANGE DIRECTION REDCAR ——
2490 XRED=XRED+1:YRED=YRED+1:RCRS=3:RETURN'       FOR LEFT
2500 XRED=XRED-1:YRED=YRED+1:RCRS=2:RETURN'       FOR UP
2510 XRED=XRED-1:YRED=YRED-1:RCRS=1:RETURN'       FOR RIGHT
2520 XRED=XRED+1:YRED=YRED-1:RCRS=4:RETURN'       FOR DOWN
2530 '
2540 '—— GO RIGHT? MYCAR ——
2550 GOSUB 3060:IF KY$<>"R" THEN RETURN'          KEY OFF
2560   XMYCAR=XMYCAR+2'                             CHANGE LOCATE
2570   IF MCRS=2 THEN C1=C1-1 ELSE C1=C1+1'                CORSE
2580 RETURN
2590 '—— GO UP? MYCAR ——
2600 GOSUB 3060:IF KY$<>"U" THEN RETURN'          KEY OFF
2610   YMYCAR=YMYCAR-2'                             CHANGE LOCATE
2620   IF MCRS=1 THEN C1=C1+1 ELSE C1=C1-1'                CORSE
2630 RETURN
2640 '—— GO LEFT? MYCAR ——
2650 GOSUB 3060:IF KY$<>"L" THEN RETURN'          KEY OFF
2660   XMYCAR=XMYCAR-2'                             CHANGE LOCATE
2670   IF MCRS=2 THEN C1=C1+1 ELSE C1=C1-1'                CORSE
2680 RETURN
2690 '—— GO DOWN? MYCAR ——
2700 GOSUB 3060:IF KY$<>"D" THEN RETURN'          KEY OFF
2710   YMYCAR=YMYCAR+2'                             CHANGE LOCATE
2720   IF MCRS=1 THEN C1=C1-1 ELSE C1=C1+1'                CORSE
2730 RETURN
2740 '—— GO LEFT OR RIGHT? MYCAR ——
2750 GOSUB 3060'                                    KEYSCAN
2760   IF KY$="L" THEN 2660'                        LEFT
2770   IF KY$="R" THEN 2560'                        RIGHT
2780 RETURN
2790 '—— GO UP OR DOWN? MYCAR ——
2800 GOSUB 3060'                                    KEYSCAN
2810   IF KY$="D" THEN 2710'                        DOWN
2820   IF KY$="U" THEN 2610'                        UP
2830 RETURN
2840 '—— GO RIGHT? REDCAR ——
2850 GOSUB 3060:IF KY$<>"6" THEN RETURN'          KEY OFF
2860   IF MCRS=2 THEN C1=C1-1:XMYCAR=XMYCAR+2:RETURN'   INSIDE
2870   C1=C1+1:XMYCAR=XMYCAR-2:RETURN'              OUTSIDE
2880 RETURN
2890 '—— GO UP? REDCAR ——
2900 CMEMO=0
2910 RETURN
2920 '—— GO LEFT? REDCAR ——
2930 CMEMO=0
2940 RETURN
2950 '—— GO DOWN? REDCAR ——
2960 CMEMO=0
```

```
2970 RETURN
2980 '---- GO LEFT OR RIGHT? REDCAR ----
2990 CMEMO=0
3000 RETURN
3010 '---- GO UP OR DOWN? REDCAR ----
3020 CMEMO=0
3030 RETURN
3040 '
3050 '---- KEYSCAN ----
3060 I0=INP(0):I1=INP(1):KY$=""'                      GET CHECK KEY ON
3070   IF I0=255 AND I1=255 THEN RETURN'               KEY OFF
3080   IF I0=239 THEN KY$="L":RETURN'                  LEFT ON
3090   IF I0=191 THEN KY$="R":RETURN'                  RIGHT ON
3100   IF I0=251 THEN KY$="D":RETURN'                  DOWN ON
3110   IF I1=254 THEN KY$="U":RETURN'                  UP ON
3120 RETURN
3130 '
3140 '▬▬ DATA AREA ▬▬
3150 '---- TITLE DATA ----
3160  DATA "█▔▔▔◣"
3170  DATA "█▁▁▁◤"
3180  DATA "█   ◣"
3190  DATA "█▁▁▁◤"
3200  DATA "◢▔▔▔◣"
3210  DATA "█   █"
3220  DATA "█   █"
3230  DATA "◥▁▁▁◤"
3240  DATA "◣   ◢"
3250  DATA "█◣▁◢█"
3260  DATA "█ ▔ █"
3270  DATA "█   █"
3280  DATA "█▔▔▔◣"
3290  DATA "█▁▁▁◤"
3300  DATA "█   ◣"
3310  DATA "█▁▁▁◤"
3320  DATA "█▔▔▔◥"
3330  DATA "█▁▁  "
3340  DATA "█    "
3350  DATA "█▁▁▁◢"
3360  DATA "█▔▔▔◣"
3370  DATA "█▁▁▁◤"
3380  DATA "█ ◥ "
3390  DATA "█▁ ◥"
3410 '
3420 '---- COURSE DATA ----
3430 DATA 00,00,00,00,07,07,00,00,00,00,03:'                  COURSE-0
3440 DATA 00,00,00,00,00,08,08,08,00,00,00,00,00,04
3450 DATA 00,00,00,00,05,05,00,00,00,00,01
3460 DATA 00,00,00,00,00,06,06,06,00,00,00,00,00,02
3470 DATA 00,00,00,99,09,09,00,00,00,99,03:'                  COURSE-1
3480 DATA 00,00,00,00,99,10,10,10,00,00,00,00,99,04
3490 DATA 00,00,00,99,09,09,00,00,00,99,01
3500 DATA 00,00,00,00,99,10,10,10,00,00,00,00,99,02
3510 DATA 00,00,99,99,09,09,00,00,99,99,03:'                  COURSE-2
3520 DATA 00,00,00,99,99,10,10,10,00,00,00,99,99,04
3530 DATA 00,00,99,99,09,09,00,00,99,99,01
3540 DATA 00,00,00,99,99,10,10,10,00,00,00,99,99,02
3550 DATA 00,99,99,99,05,05,00,99,99,99,03:'                  COURSE-3
3560 DATA 00,00,99,99,99,06,06,06,00,00,99,99,99,04
3570 DATA 00,99,99,99,07,07,00,99,99,99,01
3580 DATA 00,00,99,99,99,08,08,08,00,00,99,99,99,02
3590 '
3600 '---- ADD DATA ----
3610 DATA +2, 0:'                                    RIGHT
3620 DATA  0,-2:'                                    UP
3630 DATA -2, 0:'                                    LEFT
3640 DATA  0,+2:'                                    DOWN
3650 '---- INITIAL DOT DATA ----
3660 DATA 1,1,1,1,1,0,1,1,1,1,1:'                             COURSE-0
3670 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
3680 DATA 1,1,1,1,1,0,1,1,1,1,1
3690 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
3700 DATA 1,1,1,1,1,0,1,1,1,1,1:'                             COURSE-1
3710 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
3720 DATA 1,1,1,1,1,0,1,1,1,1,1
3730 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
3740 DATA 1,1,1,1,1,0,1,1,1,1,1:'                             COURSE-2
3750 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
3760 DATA 1,1,1,1,1,0,1,1,1,1,1
3770 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
3780 DATA 1,1,1,1,1,0,1,1,1,1,1:'                             COURSE-3
3790 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
3800 DATA 1,1,1,1,1,0,1,1,1,1,1
3810 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
```

REDCARの追撃

はじめに

マイコンに無関心な人：シラケニスト
マイコンに関心を持った人：アセリニスト
マイコン持っていない人：ナイコンニスト
マイコン買ったばかりの人：ニヤニスト
マイコン持ってる人：ジュニア　マイコンニスト
〝月刊マイコン誌〟買っている人：**マイコンニスト**
（コマーシャル，ゴメン！）

ヤカン頭のカツラをたたけば，**マイコン開化**の音がする。ネコも杓子もポンコツ親父も，マイコン，マイコン！

ままよ，我も海の子，乗り遅れるな。うらめしきマイコンよ。そしてシコシコBASIC。ああ憐れ中年世代。しかし，そのマイコン・ブームにも

「平気，平気」

と泰然自若を構えるのが，**ナイス・ミドル**だそうで。

Egg in the night ?

これ，わかります？

エグいんじゃナイ(ト)？

（ホラはフクザワ，ウソはユーキチ）

かくて**「BOMBER」に挑戦　第5章**のはじまり，はじまり。

第5章の目標設定

PRINT　〝ー〟

から始まった我々の

〝BOMBER〟

への挑戦も，そろそろ

ゴール

が見えてきました。そこでこれからの予定です。

〈第5章の目標〉

REDCARをMYCARの状況によって進路変更させる

〈第6章の目標〉

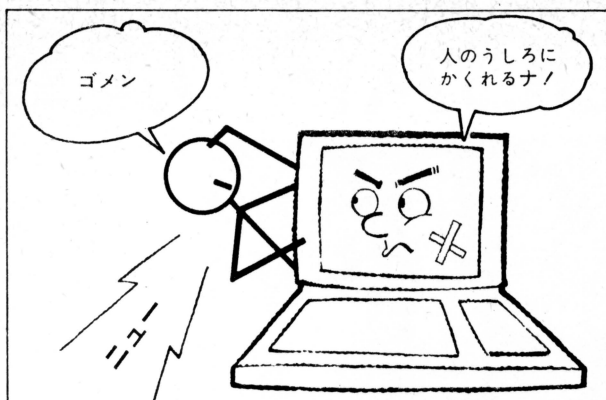
GAMEの完成

といった具合でいかがでしょう(**第3－39図**)。

それでは，いよいよ

REDCARの進路変更

に挑戦です。

《第3－39図》　今後の予定

《第３－40図》 また失敗の図

ソフトウエア生産向上のために

REDCARの進路変更
です。どうやったらいいのでしょうね？
物 真 似
日本人のお得意です。第４章で御紹介しましたように,
カーニハン ＆ プローガー
先生おっしゃるように
既存のプログラム
があれば，それを利用しましょう。使えそうなルーチンがあれば，93％利用しましょう（何だ，この中途半端な数字は？）。あくまでも**貪欲な精神**でのぞみましょう。そして
プログラミングの生産性を向上
させましょう。
我々は，すでに，
MYCARの進路変更
には成功しています。だったら，このルーチン
REDCARの進路変更
にも使えそうだとは思いませんか？
似て非なるもの
大同小異
又，いいや。少々違っていてもいいでしょう。使えないことはない。
MYCARの進路変更ルーチン
を大いに利用しましょう（横着な私は，第５章のリストのREDCARの部分は，MYCARの部分を
スクリーン・エディタ
でそっくりコピーし，修正を加えて作り上げました)。
かくて我々の合意点は，
MYCARの進路変更（第４章分）
を利用しつつ，第５章の目標である
REDCARの進路変更
を少しでも楽に作り上げていこう，ということです。
——能書きはいいから，早く始めろ（平安卿エイリアン）！

どのモジュールを変更する？

まずメインルーチンから考えてみましょう。
1660行からが，
GAME進行部
のメインルーチンが記述されています。そこでは，
① マイカーを動かす：2310行
② レッドカーを動かす：2430行
のサブルーチンが交互にCALLされてプログラムが進行して行きます。**レッドカーのコース変更**の場面は当然②の方に出てきます。そこで次に
2430行：MOVE REDCAR
を調べることになります。
2430行－2530行が，レッドカーを動かす部分です。
レッドカーは，
真っすぐ進むのか？
曲がるのか？
コースを変えるのか？
をどのように判断するのでしょうね？ それは前章までの分析のように
CRS（C３，C４）
の値で判定します。この値は，2470行で
CMEMO＝CRS（C３，C４）
のように**変数CMEMO**に記憶されますから，CMEMOの値で判定すればよいのです。すなわち

CMEMO＝
- 0：単純に前進
- 1－4：それぞれの方向に曲がる
- 5－8：コース変更（１方向）
- 9－10：コース変更（２方向）

ですから，**CMEMOの値が**
5～10
のとき，**REDCARのコース変更処理**が必要になってきます。
そこでとりあえず，

CMEMO＝5

すなわち〝右にコース変更〞する場合を例にとって説明していくことにします。

4個所のCMEMO＝5

それでは，どういうとき

CMEMO＝5

となるでしょうか？　リストの3710行－3860行の

COURSE　DATA

を御覧ください。全部で4個所

0 5

というデータが用意されていますね？　これを第3－41図で確かめると，

最も外側のコースを上に向かうとき

最も内側のコースを下に向かうとき

の二つのケースに分かれることがわかります。

REDCARが，この四つの点のいずれかを通過すると，

CMEMO＝5

となり，2490行の

ON　CMEMO　GOSUB　～

の5番目のサブルーチン（2980行）に飛び込むことになります。そこでそのサブルーチンを調べてみることにしましょう。大部焦点が絞られてきましたね。

――なにも焦点絞らんでもヨカ。ハヨGAME作って遊びましょ（平安卿エイリアン）。

《第3－41図》CMEMO＝5となるケース

コース変更可能な二つの条件

最初にその

〝GO　RIGHT？　REDCAR〞

の流れを，フローチャートで眺めておくことに致しましょう。第3－42図を御覧ください。

このサブルーチンに飛び込んできますと，最初に

コースを変える必要があるか？

チェックします。たとえば現在

REDCARとMYCAR

が**同じコース内**にいる

のであれば，REDCARはコースを変える必要がないのですから，何もしないでそのままRETURNします。

もしコースを変えるのであれば，REDCARの

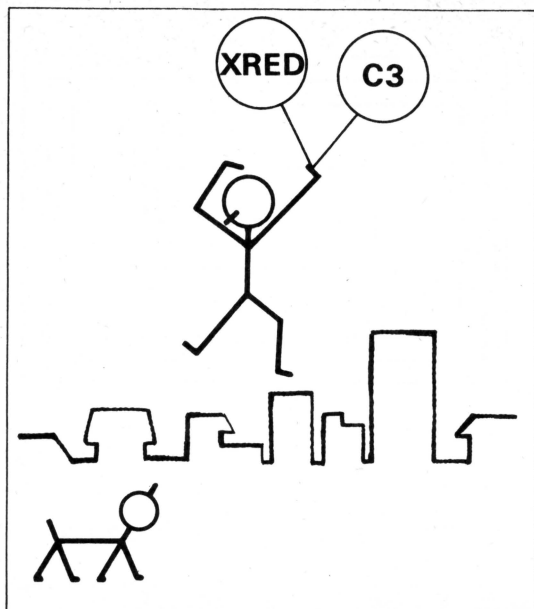
ヨコ座標：XRED

コース番号：C3

を変える作業が必要となります。もちろんヨコ座標とは，LOCATEで指定するパラメータのことですよ（第3－43図）。

以上がその流れです。それでは，もう少し具体的に説明してみましょう。

まず最初の

《第3－42図》〝GO　RIGHT？　REDCAR〞フローチャート

《第３－43図》コースを変えるには二つの要素を変える必要があります

REDCARがコースを変えては
いけない条件

というのが，二つあるのにお気付きでしょうか？　一つは，先程述べましたようにREDCARとMYCARが

同一コース上にある

ときです。**もう一つの条件**は，案外見逃しやすいかもしれません。**第３－44図**で説明致します。

図のようにREDCARが上に向かったとします。まずⒶ点に到達し，ここで進路変更可能なら（すなわちMYCARが内側にいるなら），Ⓑ点に移動します。さらにⒷ点もコースの変更可能地点なので，MYCARがそれより内側にあれば，ここでもコースの変更を行ってしまいます。これを総合すれば，

一つのインターチェンジで
二つ分のコース変更

をしてしまう，ということです。更にいえば，左右方向では

３コース分の変更をしてしまう

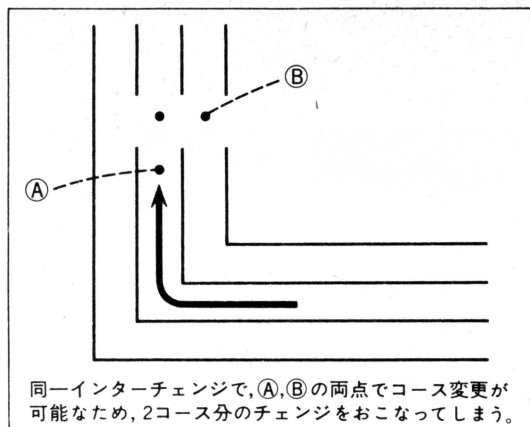
ということも有り得るわけです。これでも別におかしくはないのですが，実際にPLAYしてみると

かなり難しいGAMEになってしまう！

のが分かります。やはりREDCARの場合，

一つのインターチェンジでは
コース変更を一回にする

のが適切な難易度になると思います。

この問題を防ぐには，どうしたら良いでしょうか？

《第３－44図》1インターチェンジで2コース分のチェンジ？

もしコース変更可能かどうかを

CMEMOの値

だけで判定していたのでは，不十分です。そこでもう一つ条件を追加することになります。

第２条件の判定の仕方

もし，コース変更を１個所に限定するとしたら，

Ⓐ点，Ⓑ点（**第３－44図**）

のどちらが適切でしょうか？　そんなこと，どちらでも同じですね。それならⒶ**点**を選ぶことに致しましょう。すなわち

コース変更可能な最初の点を選ぶ！

ということです。すると，どういう条件を追加したら良いかが自然と分かってきます。

最初の点

ということは，

その一つ手前はコース変更が
不可能であった！

ということです。これはおわかりですね？

リストの2450行を御覧ください。

MEMO＝CRS（C３，C４）

となっています。これは，MEMOに一つ手前の状況を入れているところです。すなわち

MEMO：一つ手前の状況
CMEMO：現在の状況

というわけです。以上のことから，第２の条件は

MEMO＝進路変更不可能
CMEMO＝進路変更可能

の二つを満たしたときOKのサインを出せば良い，ということになります(ちなみに**第３－44図**のⒷ点の場合，

MEMOもCMEMOも

進路変更可能
ということで，第2の条件に反することになります)。
以上の考察を納得していただければ，その2条件をプログラム化した2980行は，簡単に理解できるのではないでしょうか？
——そりゃ，無理だ。(平安卿エイリアン)

一方向の場合

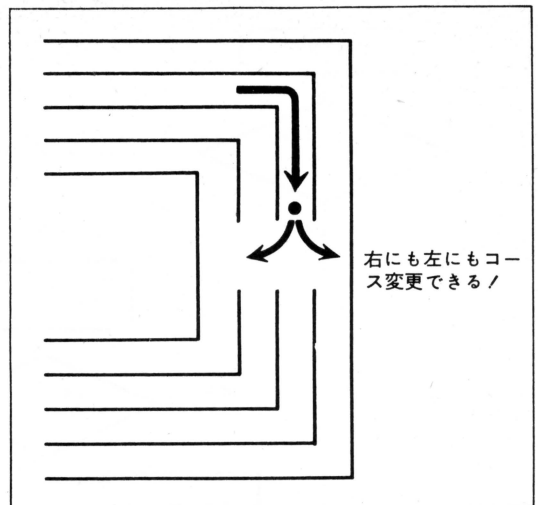
2980行の中の論理式を見ると，

C3=C1　OR　MEMO<>0

となっています。すなわち**OR**で二つの条件が結ばれているわけです。

① **第1の条件：C3=C1**
これは先に見ましたように，もし
REDCARとMYCARが同じコース
なら〝コース変更の必要〟はありませんよ，と言っているものです。

② **第2の条件：MEMO<>0**
MEMOには一つ手前の位置の情報が入っています。それが0（=単純な直進）でなければ，その直前は5～10のいずれか（インターチェンジの近くでは，0と5～10しか有り得ない！）ですから，コース変更を行っている可能性があります。したがって，この場合も〝コース変更の必要〟はありません。

以上が，コースの変更をすべきかの判定の仕方です。この二つの条件を満たしたとき，初めて
コースの変更作業
に移ることになります。
先のフローチャートで見ましたように，
二つの要素
を変更する必要があります。その一つがXREDで，+2してやります（2990行）。またコース番号C3も進路方向RCRSの値により変更します（3000行）。
以上までで，REDCARの右側へのコース変更の考察は終了します。他の
左，上，下の**一方向**への変更
もまったく同様にしてプログラム化することができます。
問題は，次の
左右方向へ進路変更可能
上下方向へ進路変更可能
の**二方向**にコースを変えられる場合です。たとえば**第3－45図**のように下に降りてきたとき，右にも左にも，もちろん真っすぐにも進むことができます。それはそ

《第3－45図》　二方向にコース変更可能

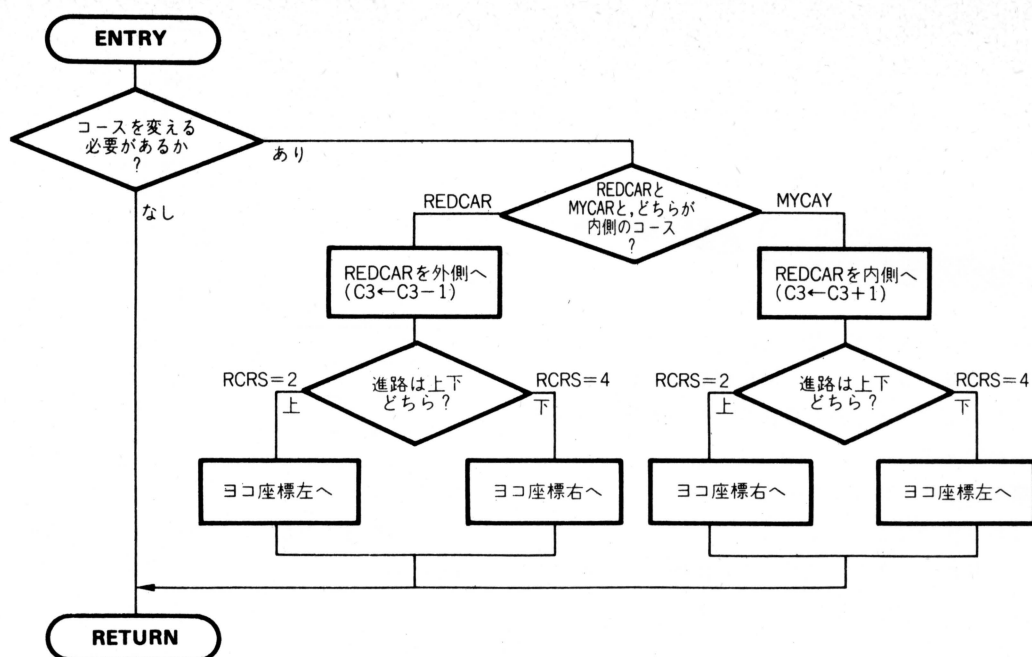
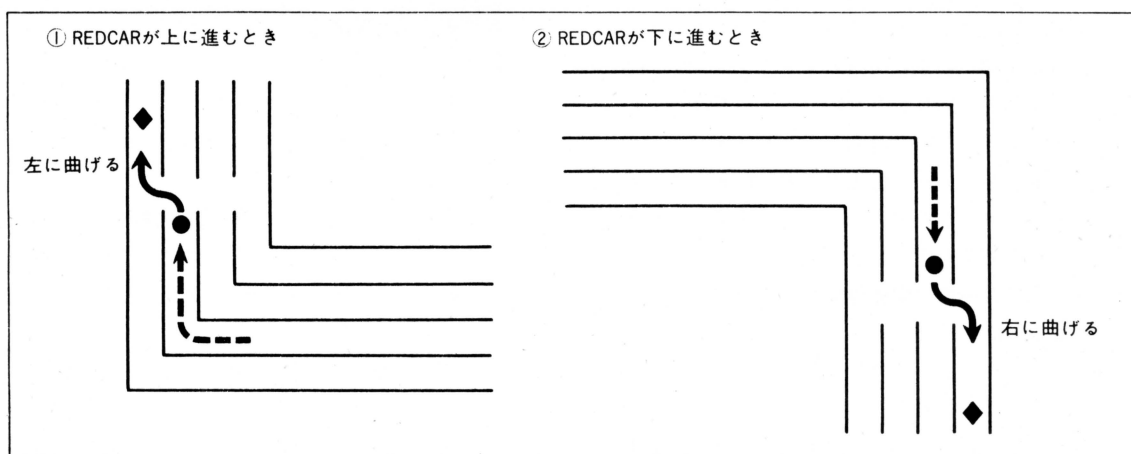
れで良いのでしょう。しかしMYCARの位置により，その進路方向を決めてやらなければなりません。どのように決めてやれば良いのでしょうね？
——乱数で決めてやれば，ヨカ。(平安卿エイリアン)

二方向の場合

進路の方向さえ決めてやれば，あとは一方向のときと同じにプログラムすることができます。したがって我々の任務は，

REDCARの進む道を決めてやる！

ことにあります。以下にその考え方を示しましょう。
リストの3210行～3250行を御覧ください。これが
左右方向への変更
のサブルーチンです。3230行は，
コースを変更すべきか？
の判定するところですから，これは特に問題ありませんね？ そこで，その部分について**第3－46図**のフローチャートで御確認ください。フローチャートによると，その後，REDCARとMYCARのコースを比べ，
REDCARの方が**内側にいる場合**
REDCARの方が**外側にいる場合**
により，**二つのケース**に分けています。ここでは，
REDCARの方が内側にいる場合
について考えてみることにします。

① **コース番号C3について**
REDCARがMYCARより内側にいるわけですから，REDCARを外側に，すなわち
C3の値を一つ小さく
してやります。

《第3－46図》 "GO LEFT OR RIGHT? REDCAR"フローチャート

《第3－47図》 REDCARが内側にいるとき

② ヨコ座標XREDについて

第3－47図のように,

　　REDCARが上に進む場合

　　REDCARが下に進む場合

の二つのケースに分けて考える必要があります。なぜなら，**第3－47図①**のように**上に進む**ときは,

　　MYCARの位置は左側

になりますから，進路を左にとります。また逆に**第3－47図②**のように**下に進む**ときは

　　MYCARの位置は右側

になりますから，進路を右にとってやります。

もう一踏ん張り

以上でREDCARの方がMYCARに比べ，内側のコースにいる場合の処理についてはおわかりいただけたことと思います。

――わかるか，アホ！（平安卿エイリアン）

他のケースについても，図を書きながら考察すればおわかりになるでしょう。また

　　上・下二方向の進路変更

についても同様に処理することができます。

さあ，私とあなたの共同作業で進めてきました

BOMBER

の製作，プログラム的にはそろそろ峠を越しましたよ。なにせ，前節までで第5章の目標である

REDCARの進路変更

にも成功しましたから，あとはこれに

得点をつけたり

の味付けをして行けば，

GAME完成！

ということになります。それは，

第6章のお楽しみ

ということにし，もう一踏ん張り，

- **MYCARとREDCARとの正面衝突のチェック**
- **バリエーションその1，その2**

に挑戦してみることにします。

衝突とバリエーション

まず，正面衝突のチェックから。

MYCARとREDCARが衝突したかを判定するのは，非常に簡単です。なぜならどちらの車も

同じ位置＝LOCATE座標が同じ

であれば衝突ですから，

{ XRED＝XMYCAR
  YRED＝YMYCAR

かどうかを調べれば良いのです。第5章までのリストでは，単に衝突したかどうかを調べ，

BEEP音

を鳴らしています。そして

MYCARの方から衝突

REDCARの方から衝突

のどちらが起こっても良いように，2400行と2520行の両方にチェック・ルーチンを入れています。

次に**バリエーションその1**。

リストの1600行を御覧ください。ここに

RGO

という新しい変数が導入されています。そして，その初期値として

RGO＝5～14

の値が設定されています。このRGO，一体何に使うのでしょう？

——そんなこと知るか！（平安卿エイリアン）

GAME進行部のメインルーチン1680行を見ると，**リスト3—15**と比べ，少し異なる点がありますね？

すなわち，

{ **RGO＝1**：REDCARを動かす
  RGO＞1：RGOから1を引く

ように変更されています。これはどういうことかというと，最初

RGO＞0

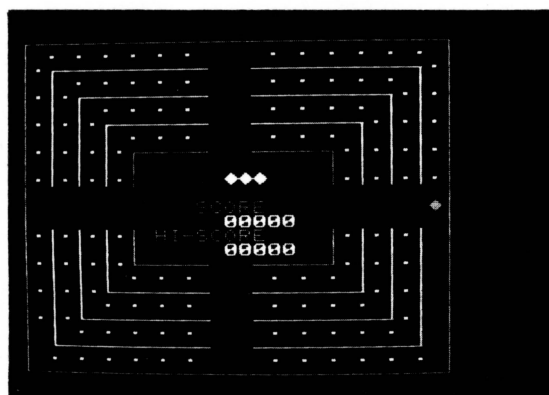
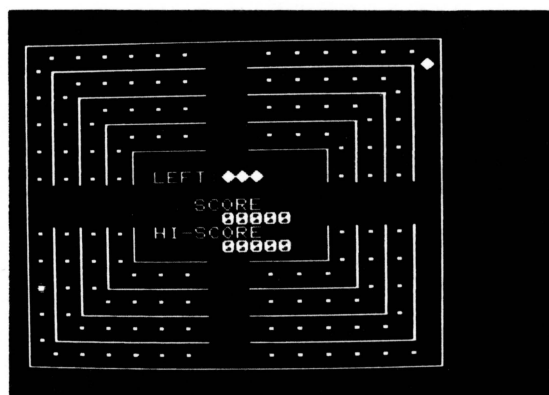
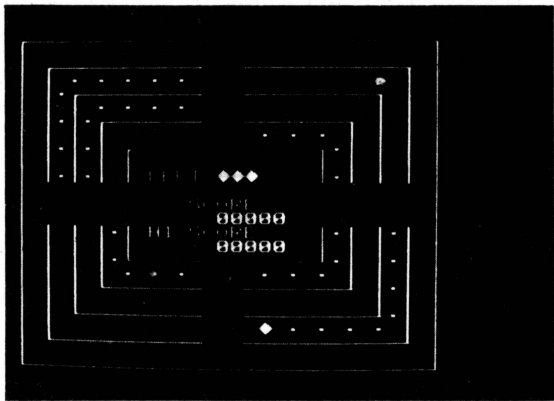
に設定されていますから，**REDCARは動きません。**そしてRGOから**1を引いて**行き，ついに

RGO＝1

になったら初めて動き出す，といったしくみです。この動きを，写真で見てみましょう。**写真14**は，プログラムをRUNさせた直後の様子です。MYCARは動き出していますが，

REDCARはまだ止まったまま

です。しばらくすると，REDCARは動き出します（**写真15**）。このように

RGO

《写真14》 RUN直後の画面

《写真15》 REDCAR発進

《**写真16**》 **MYCAR対REDCAR**

という変数を導入することにより，MYCARとREDCARの動きのタイミングがズレ，ぐっとGAMEが難しくなります。MYCARのコースをいろいろ変えると，REDCARもコースを変えてきます（**写真16**）。**写真17**は，衝突したところで，とりあえず

BEEP音

が鳴るだけです。そして**写真18**は，すべてのドットを消したところです。

バリエーションその2

バリエーションその2は，今後は逆にMYCARのためにMGO

という変数を導入しています。すなわちMYCARの方にもブレーキをかけようというわけです。

MGO＝1

のうちは動きますが，

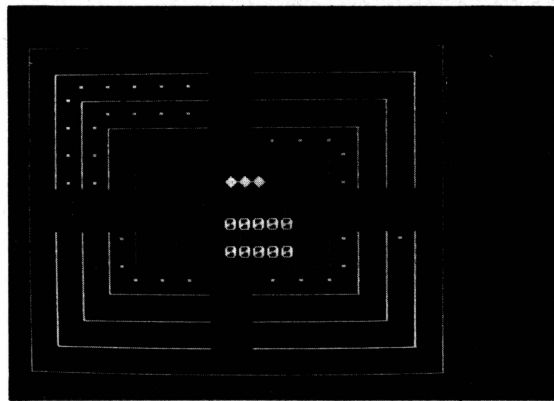
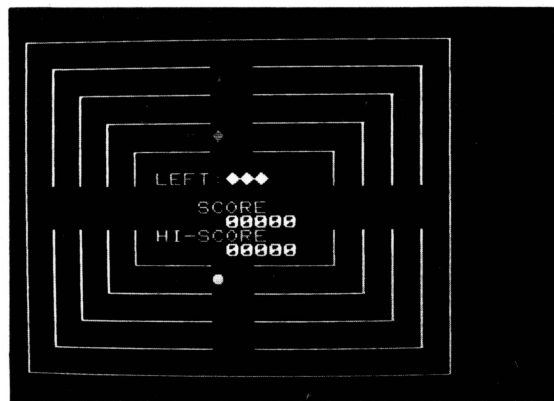
MGO＞1

になると**MYCARの動きは止まります**。つまり，バリエーションその2は，

MYCAR故障の導入！

です。この故障はいつ起きるかというと，インターチェンジで

コース変更を行わないとき
故障が起こりやすい

ように組んであります。それは，3360行を見ていただければおわかりになると思います。

《**写真17**》 **ドーン 大衝突！**

《**写真18**》 **すべてのドットを消す**

おわりに

さあ，第6章はいよいよBOMBERシリーズの最終回です。あとは，**得点**をつけ，**YCARの数を減らし**て行き，すべてクラッシュしたら，

GAME OVER！

となるように味付けしていけばよいのです。その間，**バリエーション**も追加したいですね。どんな形のGAMEが完成するか？ お楽しみ

（注，〝平安京エイリアン〟は電気音響㈱の登録商品です。）

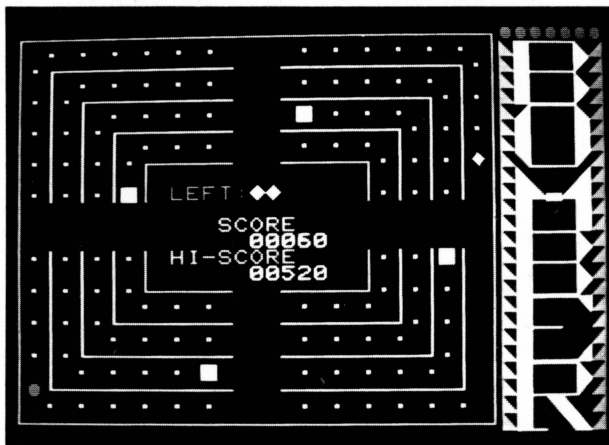
《リスト3－16》BOMBERプログラムリスト(第5章)

```
1000 '♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
1010 '♥  BOMBER  for 'INVITATION FOR GAMING'2-5 ♥
1020 '♥                << 82.2.15-X.XX >>       ♥
1030 '♥                       by K.TUKAGOSHI ♥
1040 '♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
1050 '
1060 '—— VARIABLE ——
1070 'HISCR               :ﾊｲ ｽｺｱ
1080 'LEFT                :COUNTER OF LEFT CAR
1090 'SCR                 :ｽｺｱ
1100 'XMYCAR,YMYCAR       :LOCATE MYCAR
1110 'C1,C2               :MYCAR ｺｰｽ & ｷｼﾞｭﾝﾃﾝ ｶﾗ ﾉ ｷｮﾘ
1120 'MCRS                :DIRECTION OF MYCAR     (1=ﾐｷﾞ,2=ｳｴ,3=ﾋﾀﾞﾘ,4=ｼﾀ)
1130 'CMEMO               :MEMO of MCRS(C1,C2) OR RCRS(C3,C4)
1140 'MEMO                :MEMO of OLD RCRS(C3,C4)
1150 'XRED,YRED           :LOCATE REDCAR
1160 'C3,C4               :REDCAR ｺｰｽ & ｷｼﾞｭﾝﾃﾝ ｶﾗ ﾉ ｷｮﾘ
1170 'RCRS                :DIRECTION OF REDCAR
1180 'I0,I1               :VALUE OF INP
1190 'KY$                 :VALUE OF KEYSCAN (KIY OFF="")
1200 'MGO                 :MYCAR START FLAG (1=OK,1<>NO)
1210 'RGO                 :REDCAR START FLAG (1=OK,1<>NO)
1220 '
1230 '—— DIMENSION ——
1240 'CRS(ｺｰｽ,ｷｮﾘ)        :ﾜｸ =0,1,2,3 (0=OUTSIDE,3=INSIDE)
1250 '                    :ｼﾝﾛ=0     ｾﾞﾝｼﾝ  [ﾂｷﾞ ﾊ ﾏｲｶｰ ﾉ ﾊﾞｱｲ]
1260 '                    :   =1-4  ﾎｳｺｳ ﾃﾝｶﾝ      (1=ﾐｷﾞ,2=ｳｴ,3=ﾋﾀﾞﾘ,4=ｼﾀ)
1270 '                    :   =5-8  ｷｰｽｷｬﾝ 1 ﾎｳｺｳ (5=ﾐｷﾞ,6=ｳｴ,7=ﾋﾀﾞﾘ,8=ｼﾀ)
1280 '                    :   =9,10 ｷｰｽｷｬﾝ 2 ﾎｳｺｳ (9=ｻﾕｳ,10=ｼﾞｮｳｹﾞ)
1290 '                    :   =99   ｼﾝﾛ ｶｳﾝﾀｰ=0 ﾍ
1300 '                    :                [ﾂｷﾞ ﾊ ﾚｯﾄﾞｶｰ ﾉ ﾊﾞｱｲ]
1310 '                    :   =1-4  ﾎｳｺｳ ﾃﾝｶﾝ      (1=ｳｴ,2=ﾋﾀﾞﾘ,3=ｼﾀ,4=ﾐｷﾞ)
1320 'XADD(X),YADD(Y)     :ﾍﾝｲ X,Y
1330 'DOT(ﾜｸ,ｷｮﾘ)         :0=" ",1="・"
1340 'TRACE$(1)           :0=" ",1="・"
1350 '
1360 '▬▬▬ MAIN ROUTINE ▬▬▬
1370 '—— COLD START ——
1380 DEFINT A-Z'                                  SET INTEGER
1390 HISCR=0
1400 DIM CRS(3,49),ADD(4,4),DOT(3,49)
1410   RESTORE 3710'                              READ CRS(3,49)
1420   FOR I=0 TO 3
1430     FOR J=0 TO 49
1440       READ CRS(I,J)
1450   NEXT J,I
1460   FOR I=1 TO 4'                              READ XADD(4),YADD(4)
1470     READ XADD(I),YADD(I)
1480   NEXT
1490   TRACE$(0)=" ":TRACE$(1)="・"'              SET TRACE$
1500 '
1510 '—— HOT  START ——
1520 RESTORE 3950'                                SET DOT
1530 FOR I=0 TO 3
1540   FOR J=0 TO 49
1550     READ DOT(I,J)
1560 NEXT J,I
1570 SCR=0:LEFT=3
1580 XMYCAR=29:YMYCAR=22:C1=0:C2=0 :MCRS=2'       RESET MYCAR
1590 XRED=1   :YRED=22  :C3=0:C4=35:RCRS=2'       RESET REDCAR
1600 MGO=1:RGO=INT(RND(1)*10)+5'                  GO FLAG SET
1610 WIDTH 40,25:CONSOLE 0,25,0,1:COLOR 5,32,0:PRINT CHR$(12);
1620 GOSUB 1710
1630   COLOR 7:LOCATE XMYCAR,YMYCAR:PRINT "◆";
1640   COLOR 2:LOCATE XRED,YRED:PRINT "●";
1650 '
1660 '—— GAME ——
1670 IF MGO=1 THEN GOSUB 2310 ELSE MGO=MGO-1'     MOVE MYCAR OR STOP
1680 IF RGO=1 THEN GOSUB 2430 ELSE RGO=RGO-1'     MOVE REDCAR OR STOP
1690 GOTO 1670
1700 '
1710 '▬▬▬ SUB ROUTINE ▬▬▬
1720 '—— ｶﾞﾒﾝ ——
1730 COLOR 5:PRINT CHR$(12);'                     PRINT ・
1740 FOR Y=1 TO 23
1750   X=(Y MOD 2)+1
1760   FOR X=X TO X+28 STEP 2
1770     LOCATE X,Y:PRINT "・";
```

```
1780 NEXT X,Y
1790 FOR I=0 TO 8 STEP 2'                          PRINT RECTANGLE
1800   IF (I MOD 8)=0 THEN COLOR 1 ELSE COLOR 5
1810   LOCATE I,I:PRINT "┌";
1820   Y1=I:Y2=24-I
1830   FOR X=I+1 TO 29-I
1840     LOCATE X,Y1:PRINT "─";:LOCATE X,Y2:PRINT "─";
1850   NEXT
1860   LOCATE X,Y1:PRINT "┐";:LOCATE X,Y2:PRINT "┘";
1870   X1=I:X2=30-I
1880   FOR Y=I+1 TO 23-I
1890     LOCATE X1,Y:PRINT "|";:LOCATE X2,Y:PRINT "|";
1900   NEXT
1910   LOCATE X1,Y:PRINT "└";
1920 NEXT
1930 X1=9 :X2=21:Y1=9 :Y2=15:GOSUB 2060'           CROSS OUT
1940 X1=14:X2=16:Y1=1 :Y2=23:GOSUB 2060
1950 X1=1 :X2=29:Y1=11:Y2=13:GOSUB 2060
1960 RESTORE 3440'                                 PRINT TITLE
1970 FOR Y=1 TO 24
1980   READ I$:COLOR 2:LOCATE 31,Y:PRINT "◣";:COLOR 6:PRINT I$;
1990   COLOR 2:PRINT "◢";
2000 NEXT
2010 COLOR 2:LOCATE 10,10:PRINT "LEFT:   ";:GOSUB 2130 'PRINT MESSAGE
2020 COLOR 4:LOCATE 13,12:PRINT    "SCORE";:GOSUB 2190
2030 COLOR 4:LOCATE 10,14:PRINT "HI-SCORE";:GOSUB 2220
2040 RETURN
2050 '
2060 '—— CLEAR RECTANGLE ————————
2070 ' ——— PARA IN:X1,X2,Y1,Y2 ——
2080 FOR Y=Y1 TO Y2
2090   FOR X=X1 TO X2
2100     LOCATE X,Y:PRINT " ";
2110 NEXT X,Y:RETURN
2120 '
2130 '—— PRINT LEFT ——
2140 IF LEFT=0 THEN RETURN'                        LEFT=0 THEN ﾋｮｳｼﾞ ｾｽﾞ
2150 FOR X=15 TO 14+LEFT
2160   COLOR 7:LOCATE X,10:PRINT "◆";
2170 NEXT:RETURN
2180 '
2190 '—— PRINT SCORE ——
2200 SCR$=RIGHT$("0000"+RIGHT$(STR$(SCR),LEN(STR$(SCR))-1),5)
2210 COLOR 7:LOCATE 15,13:PRINT SCR$;:RETURN
2220 '—— PRINT HI-SCORE ——
2230 HISCR$=RIGHT$("0000"+RIGHT$(STR$(HISCR),LEN(STR$(HISCR))-1),5)
2240 COLOR 7:LOCATE 15,15:PRINT HISCR$;:RETURN
2250 '—— CHANGE DIRECTION MYCAR ——
2260 XMYCAR=XMYCAR-1:YMYCAR=YMYCAR+1:MCRS=1:RETURN'     FOR RIGHT
2270 XMYCAR=XMYCAR+1:YMYCAR=YMYCAR+1:MCRS=2:RETURN'     FOR UP
2280 XMYCAR=XMYCAR+1:YMYCAR=YMYCAR-1:MCRS=3:RETURN'     FOR LEFT
2290 XMYCAR=XMYCAR-1:YMYCAR=YMYCAR-1:MCRS=4:RETURN'     FOR DOWN
2300 '
2310 '—— MOVE MYCAR ——
2320 COLOR 5:LOCATE XMYCAR,YMYCAR:PRINT " ";'      ERASE MYCAR
2330   CMEMO=CRS(C1,C2)'                           CMEMO=ｼﾝﾛ
2340   DOT(C1,C2)=0'                               ERASE DOT
2350   IF CMEMO=99 THEN C2=C2+1:GOTO 2330'         SKIP 99
2360   C2=C2+1:IF C2=50 THEN C2=0'                 1-ROUND END ?
2370   ON CMEMO GOSUB 2260,2270,2280,2290,2620,2680,2740,2800,2860,2920
2380   XMYCAR=XMYCAR+XADD(MCRS):YMYCAR=YMYCAR+YADD(MCRS)
2390 COLOR 7:LOCATE XMYCAR,YMYCAR:PRINT "◆";'      PRINT MYCAR
2400 IF XRED=XMYCAR AND YRED=YMYCAR THEN BEEP
2410 RETURN
2420 '
2430 '—— MOVE REDCAR ——
2440 COLOR 5:LOCATE XRED,YRED:PRINT TRACE$(DOT(C3,C4));'  ERASE REDCAR
2450   MEMO=CRS(C3,C4)'                            MEMO=ｷｭｳ ｼﾝﾛ
2460   C4=C4-1:IF C4<0 THEN C4=49'                 ROUND END ?
2470   CMEMO=CRS(C3,C4)'                           CMEMO=ｼﾝﾛ
2480   IF CMEMO=99 THEN C4=C4-1:GOTO 2470'         SKIP 99
2490   ON CMEMO GOSUB 2570,2560,2590,2580,2980,3040,3100,3160,3220,3280
2500   XRED=XRED+XADD(RCRS):YRED=YRED+YADD(RCRS)
2510 COLOR 2:LOCATE XRED,YRED:PRINT "●";'          PRINT REDCAR
2520 IF XRED=XMYCAR AND YRED=YMYCAR THEN BEEP
2530 RETURN
2540 '
2550 '—— CHANGE DIRECTION REDCAR ——
```

```
2560 XRED=XRED+1:YRED=YRED+1:RCRS=3:RETURN'        FOR LEFT
2570 XRED=XRED-1:YRED=YRED+1:RCRS=2:RETURN'        FOR UP
2580 XRED=XRED-1:YRED=YRED-1:RCRS=1:RETURN'        FOR RIGHT
2590 XRED=XRED+1:YRED=YRED-1:RCRS=4:RETURN'        FOR DOWN
2600 '
2610 '—— GO RIGHT? MYCAR ——
2620 GOSUB 3330:IF KY$<>"R" THEN RETURN'           KEY OFF
2630   XMYCAR=XMYCAR+2'                            CHANGE LOCATE
2640   IF MCRS=2 THEN C1=C1-1 ELSE C1=C1+1'               CORSE
2650 RETURN
2660 '
2670 '—— GO UP? MYCAR ——
2680 GOSUB 3330:IF KY$<>"U" THEN RETURN'           KEY OFF
2690   YMYCAR=YMYCAR-2'                            CHANGE LOCATE
2700   IF MCRS=1 THEN C1=C1+1 ELSE C1=C1-1'               CORSE
2710 RETURN
2720 '
2730 '—— GO LEFT? MYCAR ——
2740 GOSUB 3330:IF KY$<>"L" THEN RETURN'           KEY OFF
2750   XMYCAR=XMYCAR-2'                            CHANGE LOCATE
2760   IF MCRS=2 THEN C1=C1+1 ELSE C1=C1-1'               CORSE
2770 RETURN
2780 '
2790 '—— GO DOWN? MYCAR ——
2800 GOSUB 3330:IF KY$<>"D" THEN RETURN'           KEY OFF
2810   YMYCAR=YMYCAR+2'                            CHANGE LOCATE
2820   IF MCRS=1 THEN C1=C1-1 ELSE C1=C1+1'               CORSE
2830 RETURN
2840 '
2850 '—— GO LEFT OR RIGHT? MYCAR ——
2860 GOSUB 3330'                                   KEYSCAN
2870   IF KY$="L" THEN 2750'                       LEFT
2880   IF KY$="R" THEN 2630'                       RIGHT
2890 RETURN
2900 '
2910 '—— GO UP OR DOWN? MYCAR ——
2920 GOSUB 3330'                                   KEYSCAN
2930   IF KY$="D" THEN 2810'                       LEFT
2940   IF KY$="U" THEN 2690'                       RIGHT
2950 RETURN
2960 '
2970 '—— GO RIGHT? REDCAR ——
2980 IF C3=C1 OR MEMO<>0 THEN RETURN
2990   XRED=XRED+2'                                CHANGE LOCATE
3000   IF RCRS=2 THEN C3=C3+1 ELSE C3=C3-1'               CORSE
3010 RETURN
3020 '
3030 '—— GO UP? REDCAR ——
3040 IF C3=C1 OR MEMO<>0 THEN RETURN
3050   YRED=YRED-2'                                CHANGE LOCATE
3060   IF RCRS=1 THEN C3=C3-1 ELSE C3=C3+1'               CORSE
3070 RETURN
3080 '
3090 '—— GO LEFT? REDCAR ——
3100 IF C3=C1 OR MEMO<>0 THEN RETURN
3110   XRED=XRED-2'                                CHANGE LOCATE
3120   IF RCRS=2 THEN C3=C3-1 ELSE C3=C3+1'               CORSE
3130 RETURN
3140 '
3150 '—— GO DOWN? REDCAR ——
3160 IF C3=C1 OR MEMO<>0 THEN RETURN
3170   YRED=YRED+2'                                CHANGE LOCATE
3180   IF RCRS=1 THEN C3=C3+1 ELSE C3=C3-1'               CORSE
3190 RETURN
3200 '
3210 '—— GO LEFT OR RIGHT? REDCAR ——
3220 IF C3=C1 OR MEMO<>0 THEN RETURN
3230   IF C3>C1 THEN C3=C3-1:IF RCRS=2 THEN XRED=XRED-2  ELSE XRED=XRED+2
3240   IF C3<C1 THEN C3=C3+1:IF RCRS=2 THEN XRED=XRED+2  ELSE XRED=XRED-2
3250 RETURN
3260 '
3270 '—— GO UP OR DOWN? REDCAR ——
3280 IF C3=C1 OR MEMO<>0 THEN RETURN
3290   IF C3>C1 THEN C3=C3-1:IF RCRS=1 THEN YRED=YRED-2  ELSE YRED=YRED+2
3300   IF C3<C1 THEN C3=C3+1:IF RCRS=1 THEN YRED=YRED+2  ELSE YRED=YRED-2
3310 RETURN
3320 '
3330 '—— KEYSCAN ——
```

```
3340 I0=INP(0):I1=INP(1):KY$=""'                         GET CHECK KEY ON
3350 IF I0<>255 OR I1<>255 THEN 3370'                    KEY ON
3360   IF RND(1)*100<95 THEN RETURN ELSE MGO=INT(RND(1)*10)+1:RETURN'   STOP
3370   IF I0=239 THEN KY$="L":RETURN'                    LEFT ON
3380   IF I0=191 THEN KY$="R":RETURN'                    RIGHT ON
3390   IF I0=251 THEN KY$="D":RETURN'                    DOWN ON
3400   IF I1=254 THEN KY$="U":RETURN'                    UP ON
3410 '
3420 '▬▬▬ DATA AREA ▬▬▬
3430 '—— TITLE DATA ——
3440  DATA "█▔▔▔◣"
3450  DATA "█___◤"
3460  DATA "█   ◣"
3470  DATA "█___◤"
3480  DATA "◢▔▔▔◣"
3490  DATA "█   █"
3500  DATA "█   █"
3510  DATA "◥___◤"
3520  DATA "◣   ◢"
3530  DATA "█◣_◢█"
3540  DATA "█ ▔ █"
3550  DATA "█   █"
3560  DATA "█▔▔▔◣"
3570  DATA "█___◤"
3580  DATA "█   ◣"
3590  DATA "█___◤"
3600  DATA "█▔▔▔◥"
3610  DATA "█___ "
3620  DATA "█    "
3630  DATA "█___◢"
3640  DATA "█▔▔▔◣"
3650  DATA "█___◤"
3660  DATA "█ ◥ "
3670  DATA "█_ ◥"
3690 '
3700 '—— COURSE DATA ——
3710 DATA 00,00,00,00,07,07,00,00,00,00,03:'                          COURSE-0
3720 DATA 00,00,00,00,00,08,08,08,00,00,00,00,00,04
3730 DATA 00,00,00,00,05,05,00,00,00,00,01
3740 DATA 00,00,00,00,00,06,06,06,00,00,00,00,00,02
3750 DATA 00,00,00,99,09,09,00,00,00,99,03:'                          COURSE-1
3760 DATA 00,00,00,00,99,10,10,10,00,00,00,00,99,04
3770 DATA 00,00,00,99,09,09,00,00,00,99,01
3780 DATA 00,00,00,00,99,10,10,10,00,00,00,00,99,02
3790 DATA 00,00,99,99,09,09,00,00,99,99,03:'                          COURSE-2
3800 DATA 00,00,00,99,99,10,10,10,00,00,00,99,99,04
3810 DATA 00,00,99,99,09,09,00,00,99,99,01
3820 DATA 00,00,00,99,99,10,10,10,00,00,00,99,99,02
3830 DATA 00,99,99,99,05,05,00,99,99,99,03:'                          COURSE-3
3840 DATA 00,00,99,99,99,06,06,06,00,00,99,99,99,04
3850 DATA 00,99,99,99,07,07,00,99,99,99,01
3860 DATA 00,00,99,99,99,08,08,08,00,00,99,99,99,02
3870 '
3880 '—— ADD DATA ——
3890 DATA +2, 0:'                                        RIGHT
3900 DATA  0,-2:'                                        UP
3910 DATA -2, 0:'                                        LEFT
3920 DATA  0,+2:'                                        DOWN
3930 '
3940 '—— INITIAL DOT DATA ——
3950 DATA 1,1,1,1,1,0,1,1,1,1,1:'                        COURSE-0
3960 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
3970 DATA 1,1,1,1,1,0,1,1,1,1,1
3980 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
3990 DATA 1,1,1,1,1,0,1,1,1,1,1:'                        COURSE-1
4000 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
4010 DATA 1,1,1,1,1,0,1,1,1,1,1
4020 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
4030 DATA 1,1,1,1,1,0,1,1,1,1,1:'                        COURSE-2
4040 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
4050 DATA 1,1,1,1,1,0,1,1,1,1,1
4060 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
4070 DATA 1,1,1,1,1,0,1,1,1,1,1:'                        COURSE-3
4080 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
4090 DATA 1,1,1,1,1,0,1,1,1,1,1
4100 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
```

仮の最終回

はじめに

朝日が登る頃
社長さんが帰る
ア〜〜いい気持ち！

昭和30年代の半ば，世にこんな楽しい唄が流行していました。どなたか御存知ありませんか？マア，「ＢＯＭＢＥＲ」に挑戦シリーズ，にぎやかに幕を閉じるのにうってつけのオープニングですね？――ワ・タ・シの独断。

栄光への歩み

第６章は

「BOMBER」の最終回

です。何？　タイトルに「仮の――」と書いてある？マ，気にしないでください。

ここで第１章からの足跡を振り返ってみましょう（**第３－48図**）。

まず我々が挑戦したのが，

ＧＡＭＥの画面

を作ることでした。我々は，**困難を分割**し，複雑なテレビ画面も

PRINT　"チョメチョメ"

に帰着できることを知りました。そして早５か月。

マイカーの動かし方

章	内　容
1	オリエンテーション GAMEエリアの製作
2	GAME仕様の分析 マイカーの動かし方の研究
3	SKIPマークの導入 レッドカーの登場 ドットの消し方
4	キー入力について マイカーの進路変更の仕方
5	相手の状況判断 レッドカーの追従の仕方

《第３－48図》"BOMBER"の製作の歩み

進路変更の仕方

ドットの消し方

等々，少しずつゲーム作りを進めてきました。そして幾多の困難（大ゲサ！）を乗り越え，

ようやく

GAME完成へ！

とこぎつけてきたわけです。

遊び方

ここではGAME製作の前に，先にその

完成版のプログラム

を走らせてみることに致します。

リストを御覧ください。これが私の**“参考出品”**です。RUNしてみましょう。トテチテター！ **第3－49図**がRUN直後の様子です。まだ何もゲームをしていませんから，

HI－SCORE＝0点

SCORE＝0点

となっています。このリストは，毎度申し上げますようにあくまでも参考出品ですから，とくに飾り立てることはしていません。本来は

デモンストレーション

をつけ，

GAMEの説明

も入れるべきでしょうね。その方がグッとGAMEが楽しくなり，また親切でもあります。ここらあたりについては，他人様の作品を良く研究してあなた自身くふうしてみてください。またささやかながら私のいくつかのGAMEが参考になるかもしれません（ならないかもしれません――アシカラズ）。

```
HI-SCORE: 0

   SCORE: 0

     1————GAME START
     2————GAME END
                ..........HOW ??
```

《第3－49図》 RUN直後の様子

第3－50図の状態で

GAMEを始めたいとき…… [1] **のキー**

GAMEをやめたいとき…… [2] **のキー**

を押してください。他のキーを押しても当然無視されます。

第3－50図は，[1] **のキー**を押してGAMEをスタートした直後です。**数秒間この状態が続きます**。この間，ゲーム進行にそなえて呼吸を整えておいてください。その間CPUの側では，**変数の初期化**をおこなっています。

第3－50図を使って**TV画面の見方**を説明しておきます。画面左側が，**GAMEエリア**。そして中央部が，**メッセージ・エリア**です。メッセージ・エリアは次のようになっています。

LEFT：MYCARの残り数。最初は3台あります。しかし画面には，2台しか表示されていません。これは，すぐに1台がGAMEエリアの右下に表示されGAME進行に使われるからです。すなわち正確に言えば，LEFTには

MYCAR予備の残り数

が表示される，というわけです。

HI－SCORE：最高点

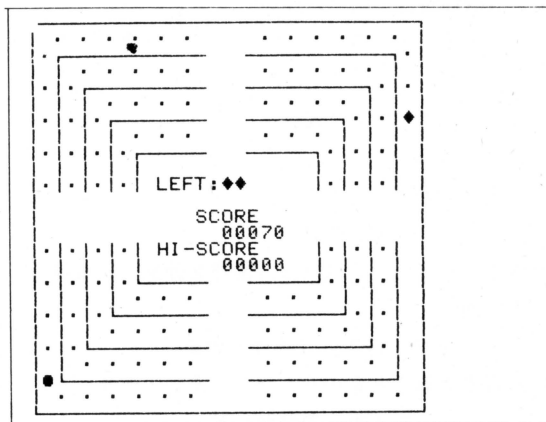
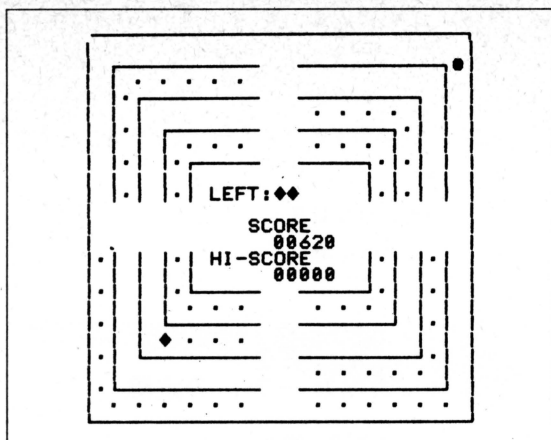
SCORE：言わずと知れた得点です

数秒後，**第3－51図**のようにMYCARが動き出し始めます。まわり方は左まわりで，動いたあと**ドット・**を消して行きます。

ドット・を一つ消す毎に10点

が加算されます。**第3－51図**では，ドットを七つ消していますから，

SCORE＝00070

《第3－50図》 キー1を押してGAMEをスタートさせる

《第3－51図》 MYCARが動き出す

《第3－52図》たくみにREDCARをよける

《第3－53図》画面は最初に戻って

になっています。

最初の数秒間は，**REDCARは，止まったままで**す。しばらくすると右まわりに動いてきますから，MYCARのコースを変えてREDCARをよけてください。私のプログラムでは，

キー［4］：**左のコース**
キー［6］：**右のコース**
キー［8］：**上のコース**
キー［2］：**下のコース**

のようにコースを変えることができます。**第3－52図**は，たくみにREDCARをよけているところです。得点は，620点に達しています。

REDCARと

正面衝突！

すると，MYCARが一つ減ります。したがって

LEFTの◆

も一つ減ります。**画面は第3－52図の状態に戻ってゲーム再開**です。ただし◆の数は，一つ減っていますが。**第3－53図**は正面衝突後，再びMYCARが動き出したところです。**第3－54図**では，さらにMYCARが一台減ったところです。もう予備がありません。これ以上衝突するとGAME　OVERになってしまいます。得点はすでに2150点に達しています。頑張ってますね。

三台ともMYCARがやられると，画面は**RUN直後の状態**に戻ります（**第3－55図**)。この場合，今のゲームで2750点をとったことになります。また今までの最高点は2100点ですから，**新記録**が出たことになります。この状態で

GAMEを始めたいとき……［1］のキー
GAMEをやめたいとき……［2］のキー

《第3－54図》もう残りがない

《第3－55図》ついにGAME OVER！

を押せば良いのは，先に述べたとおりです。

以上のところまで，この章で作っていきたいと思います。エ？　もう疲れました？

数の整形術

それではプログラムの説明に移りましょう。当然第5章のプログラムを**変更の**上，新しい部分を**追加**してあります。そして最後に整形のため**リナンバー**がかけてあります。したがってあなたも御自分のプログラムにリナンバーをかけながら，照らし合わせていただくと良いでしょう。

それでは，まず得点の表示についてですが，得点が30点のとき，

SCORE：30――――①

SCORE：00030――――②

とどちらが簡単でしょうか？

〈その1〉

BASICでは②よりも①のように表示させる方が簡単ですが，マシン語の場合は，②の方が楽です。もっとも文字列表示ルーチンを持っている場合は，どちらでも同じですが。

〈その2〉

BASICの行番号の表示の仕方は，100％①の形をとっていますが，②の方が合理的だと思いませんか？　たとえば，

```
10 PRINT "┌────┐"
20 PRINT "│    │"
30 PRINT "└────┘"
```

とあれば箱を書こうとしているのは，すぐに分かります。しかし運が悪いと，

```
90 PRINT "┌───┐"
100 PRINT "│   │"
110 PRINT "└───┘"
```

のようにズレて，**非常に見にくいプログラムになってしまいます。しかし②のように表示してくれれば，**

```
00090 PRINT "┌────┐"
00100 PRINT "│    │"
00110 PRINT "└────┘"
```

のようにきちんと揃います。

現在，BASICの行番号表示が①の形をとっているのは，**行番号の表示を単純に“数の出力ルーチン”でおこなっているため**と思われます。

“数の出力ルーチン”は，**“文字列表示ルーチン”のバッファに“数のASCIIコード”を転送しているだけ**ですから，自然に左詰めで表示されてしまいます。将来BASICインタプリタの設計をする人は，ここらあたりの改善をお願いしたいですね。

なお私が1000からリナンバーをかけているのは，**行番号の桁数を4ケタに揃えるため**です。御参考までに。

〈その3〉

もう一つおまけです。

数を右詰めで表示する場合（例として4桁），

0030　――――①

　30　――――②

①のように前に0が入る形と，②のように余計な0の入らない形があります。①を②のように表わすことを

ゼロサプレス

(zero suppress)

と言います。

“REPLAYルーチンの定義”

そこで**第3－49図**と**第3－55図**を御覧ください。まったく同じ画面ですね？　この二つの部分は，プログラム上でも同じサブルーチン

1700行――1800行

を用いています。1700行をみると，

――GAME　REPLAY――

とサブルーチンのタイトルが書いてあります。そうです。もともとこのルーチンは，

GAME　OVER

の際に今やったゲームの得点，またその得点が今までの最高点を越えているのか否かを表示し，その上で

もう**GAMEをやめる**のか

それとも**GAMEを続ける**のか

の態度を得るためのルーチンだったのです。しかし，

得点
最高点 } ＝0

にしておけば，このルーチンをGAMEスタート直後に使ってもおかしくありませんね？

以上をふまえた上で，このルーチンを

“REPLAYルーチン”

と呼ぶことにします。そこで**第3－56図**を御覧ください。

ストラクチャード・プログラミング？

これが，**プログラム全体を表わすフローチャート**です。番号をふっておきましたから，それと合わせて以下をお読みください。

① プログラムの**START**です。

② **“REPLAYルーチン”**，おわかりですね。最初は，たぶんGAMEをするでしょうね。でもしないかもしれません。いずれにせよ次の③に進みます。

③ **分岐点**です。GAMEをするならそのまま下に降りて④に進みます。もしGAMEをやめるなら，⑥

に分岐します。

④ **GAME**が展開されます。

GAME　OVER

になると，次の⑤に進みます。

⑤ 再び **"REPLAYルーチン"** です。得点類を表示し，再度GAMEを続けるかの態度を得ます。そして③に戻ります。

⑥ ③→④→⑤ をグルグル回った結果（あるいは④，⑤を一度も通過しないかもしれません），もうGAMEをやらないときにここにきます。ここには，

END

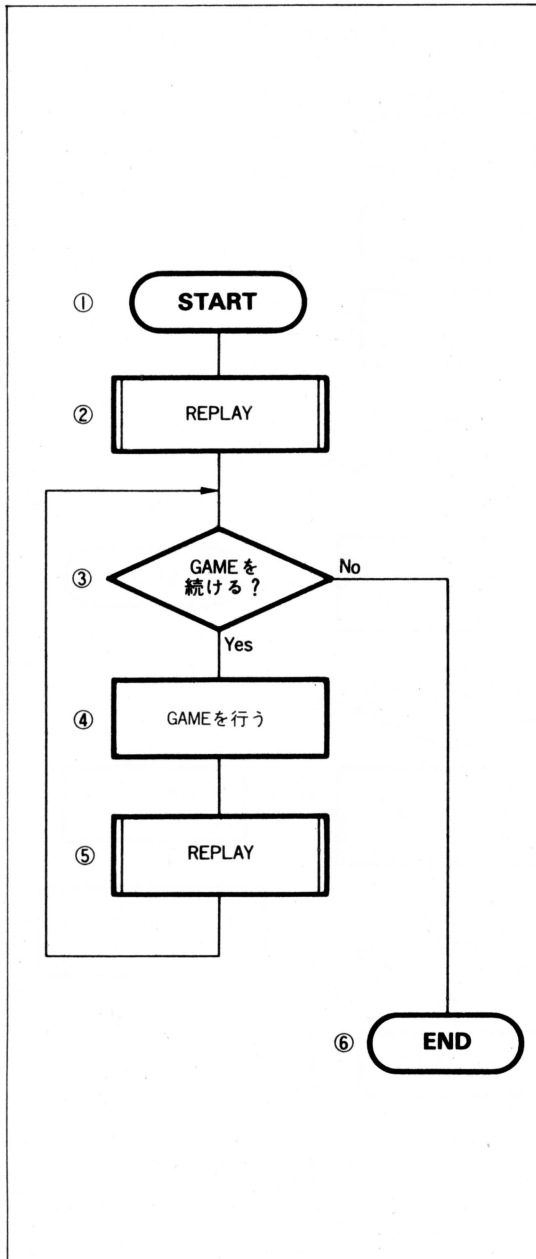
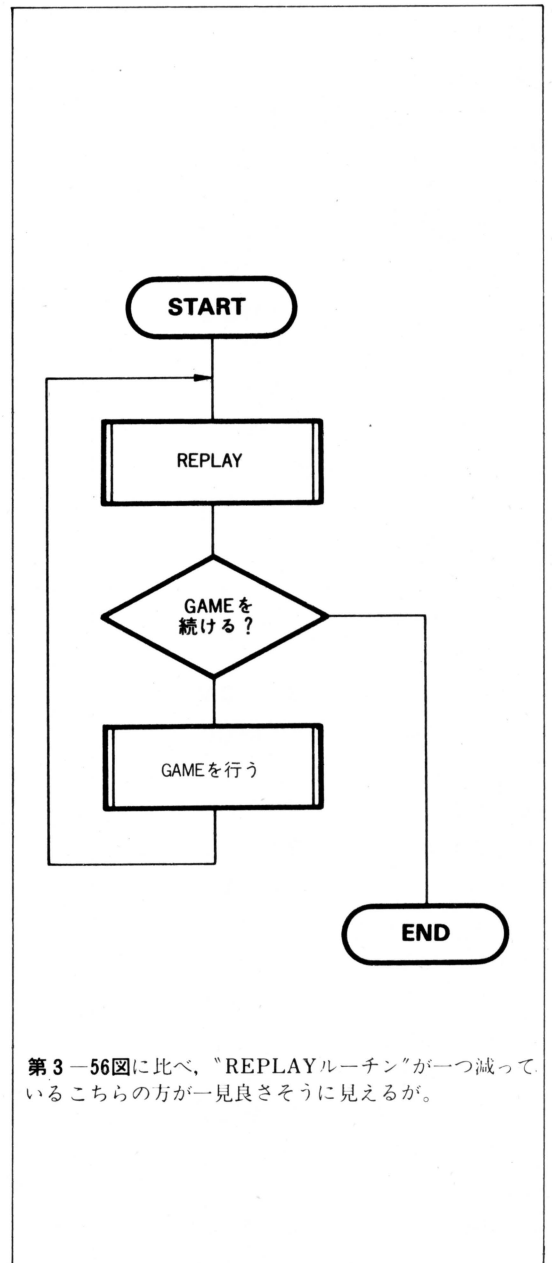
の命令が書いてありますから，当然プログラムが終了することになります。

と，ここまで読んでこられたあなたは，

「馬鹿らしい！」

とか，ある疑問を持ったかもしれません。

たぶんあなたの持たれた疑問は，次の通りでしょう。

すなわち，**第3―56図**よりは**第3―57図**の方がbetterではないか？　なぜなら**第3―57図**の方が，

《第3―56図》プログラムの流れOVER！

《第3―57図》こちらの方がbetter ？

"REPLAYルーチン"の数が少ないし，それだけ**省メモリ**だから。そう思ったあなたは，多分反省する必要があるでしょう（詳しくは，月刊マイコン連載中のGAMINGへの招待「ストラクチャード・プログラミング」を御覧ください）。

GAMEの四つのレベル

次にGAMEの**進行を司る管理の仕方**を見てみましょう。

プログラム全体をGAMEの状況から見ると，次の**4**段階に分かれます。

〈レベル0〉

GAME進行中の状態。

〈レベル1〉

MYCARとREDCARとの**正面衝突**が起こったとき。このときは，GAME OVERではありませんが，GAMEを一時停止させます。そして

MYCARを初期位置に
REDCARを初期位置に
ドットを最初の状態に

してやる必要があります。すなわち

得点のクリア
MYCARの残りの数を初期値に } (※)

戻さない他は，**GAME OVER後の処理とまったく同じ**になります。

〈レベル2〉

GAME OVERのとき。このときは，"REPLAYルーチン"をCALLして再GAMEを行なうか調べます。再GAMEをするときは，(※)の処理をした後，

最高点のチェック

をし，〈レベル1〉の状態に戻してやります。

〈レベル3〉

第3－49図，第3－55図において 2 のキーを押した状態です。**プログラム終了**ですから使うコマンドは

END

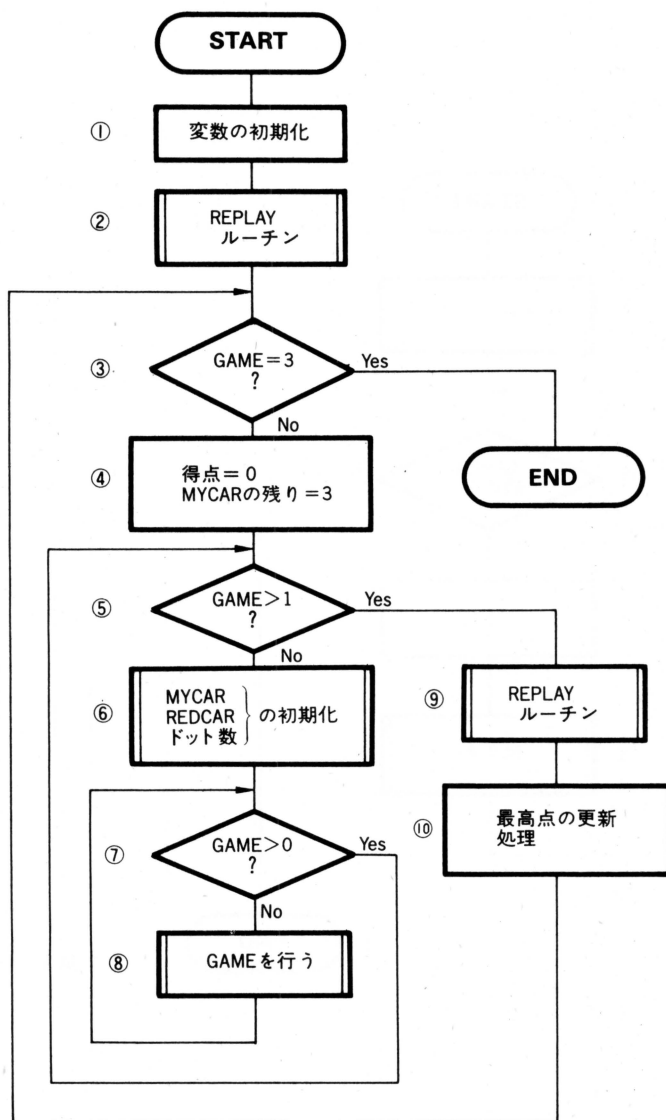
だけです。

変数GAMEでゲームを管理

プログラムは以上のように**四つのレベル**から成立しています。プログラムの中でその四つのレベルを管理するのに

変数GAME

を使って区別しています。たとえば

GAME＝2

《第3－58図》四つのレベルを管理する

なら〈レベル2〉にあるわけです。

以上の予備知識をもとに，更に**第3－58図**のフローチャートも参考にプログラムの説明をしていくことにします。

まず1430行で

HISCR＝0

で最高点を0にしています。これが**第3－58図①**の

変数の初期化

ですね。続いて1560行で“REPLAYルーチン”をCALLしています。②ですね。

“REPLAYルーチン”をみてみましょう。1700行－1800行です。

1710行—1760行は，**第3－49図**の**画面**を**表示**しているところです。1770行は，変数Iに態度を入力しています。もちろん

INPUT　GAME

でもかまいません。結局1770行－1800行で

ゲームをする：GAME＝0
ゲームをしない：GAME＝3

を得ているわけです。

1570行が③の部分です。GAME＝3のときは，1660行からのENDルーチンに入ります。と言っても

END文

の1行しかありません。本来はこの部分も手抜きせずに面白い**ENDデモをすべき**でしょうね。それについては，あなたにおまかせします。

1580行で

LEFT＝3：MYCARの残り数
SCR＝0：得点

の初期を行っています。④の部分ですね。

1590行が⑤の部分です。

GAME＞1

とありますが，実際には

GAME＝0：GAMEを行う
GAME＝2：GAME　OVER

の場合しかこの部分を通りません。

GAME＝2：GAME　OVER

のときは，THEN以下が実行されます。すなわち⑨，⑩ですね。

1600行で⑥のサブルーチンがCALLされます。1860行－1990行です。1870行で

GAME＝0：ゲームを行える状態にする。
DOT＝129：ドットを初期値にする

を行っています。この部分については二点ほど補足をしておきましょう。

ドットの数が合わない？

〈補足その1〉

1780行ですでにGAME＝0

になっているのに，なぜわざわざ1870行でもう一度

GAME＝0

にセットしているのか？　答は，ゲーム進行中にMYCARとREDCARが正面衝突したとき

GAME＝1

でこのルーチンに飛び込んでくるからです。ですからDOTの数を初期値に戻してやり，**もう一度GAMEが出来るようにGAME＝0にリセット**しているのです。

〈補足その2〉

第3－50図でドット・の数を数えてみてください。全部で128ありますね？　それにもかかわらず1870行で

DOT＝129

にセットしたのはなぜでしょう？

その答は少々複雑です。ドットの数はゲーム進行中，**どんなときに変化する**でしょうか？　MYCARがドット・を消したときですね。その処理をどの部分でやっているかお気付きでしょうか？

MYCARがDOTを消すと，

得点をカウント・アップ

しなければなりません。すなわち

2490行－2530行

PRINT　SCORE

ルーチンがCALLされます。逆に言えば，このルーチンの中に

ドットの数を減らす

処理を書いておいても良いわけです。そうすれば自動的にMYCARがドットを消したとき，ドットの数が減ってくれます。

リストを御覧ください。“得点表示ルーチン”の中の2520行で

DOT＝DOT－1

とやってドットの数を減らしていますね？

ところでこのようにすると，一つ**不合理**が生じます。と言うのは，**最初にゲームの画面を描くときにこの**

PRINT　SCORE

ルーチンをCALLしているからです。つまり

何もゲームをしていないうちに

ドットの数が一つ減ってしまう

というわけです。これを防ぐには，あらかじめドットの数を一つ増やしておけば良いのです。これが1870行で

DOT＝129

にしている理由です。

完成！

“SET CAR START POSITION”のその他の部分については，すでに解析が終っていますね？

⑦，⑧の部分が，**GAME進行中**繰り返される部分です。プログラムで言えば

1610行－1640行

に当たります。これは

GAME＞0

になるまで続きます。**GAME本体**については，これまた第5章までで解析を済ませていますね？——と言うことは，我々はついに

BOMBER

のすべてにわたって解析を終えたことになります。

長い長〰〰い道のり

本当に御苦労様でした。

おわりに

しかしまだ油断をしては，いけません。本プログラムには，

デモンストレーションがない！

GAMEの説明がない！

等の手抜きがしてあります。そこで第7章は，この部分にスポットを当ててみることに致しましょう。それが真の意味での

〈BOMBERに挑戦〉最終回

となるでしょう。

《リスト3—17》BOMBERプログラムリスト（第6章）

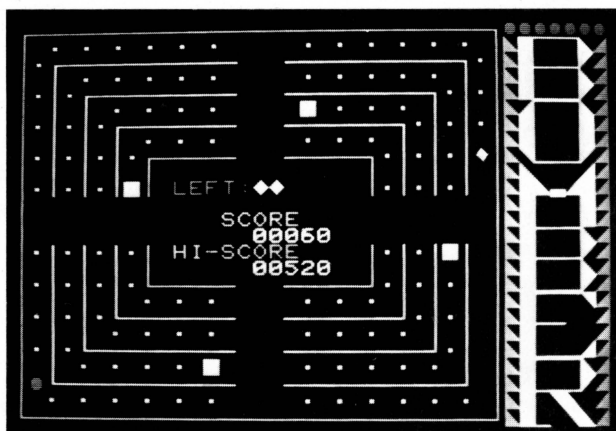
```
1000 '♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
1010 '♥  BOMBER  for 'INVITATION FOR GAMING'2-6 ♥
1020 '♥                <<  82.2.15-X.XX >>     ♥
1030 '♥                        by K.TUKAGOSHI ♥
1040 '♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
1050 '
1060 '—— VARIABLE ——
1070 'HISCR             :ﾊｲ ｽｺｱ
1080 'LEFT              :COUNTER OF LEFT CAR
1090 'SCR               :ｽｺｱ
1100 'XMYCAR,YMYCAR     :LOCATE MYCAR
1110 'C1,C2             :MYCAR ｺｰｽ & ｷｼﾞｭﾝﾃﾝ ｶﾗ ﾉ ｷｮﾘ
1120 'MCRS              :DIRECTION OF MYCAR      (1=ﾐｷﾞ,2=ｳｴ,3=ﾋﾀﾞﾘ,4=ｼﾀ)
1130 'CMEMO             :MEMO of MCRS(C1,C2) OR RCRS(C3,C4)
1140 'MEMO              :MEMO of OLD RCRS(C3,C4)
1150 'XRED,YRED         :LOCATE REDCAR
1160 'C3,C4             :REDCAR ｺｰｽ & ｷｼﾞｭﾝﾃﾝ ｶﾗ ﾉ ｷｮﾘ
1170 'RCRS              :DIRECTION OF REDCAR
1180 'I0,I1             :VALUE OF INP
1190 'KY$               :VALUE OF KEYSCAN (KIY OFF="")
1200 'MGO               :MYCAR START FLAG (1=OK,1<>NO)
1210 'RGO               :REDCAR START FLAG (1=OK,1<>NO)
1220 'DOT               :LEFT OF DOT
1230 'GAME              :GAME FLAG (0=GAME ON,1=CRASH,2=GAME OVER)
1240 '
1250 '—— DIMENSION ——
1260 'CRS(ｺｰｽ,ｷｮﾘ)      :ﾜｸ =0,1,2,3 (0=OUTSIDE,3=INSIDE)
1270 '                  :ｼﾝﾛ=0    ｾﾞﾝｼﾝ  [ﾂｷﾞ ﾊ ﾏｲｶｰ ﾉ ﾊﾞｱｲ]
1280 '                  :    =1-4  ﾎｳｺｳ ﾃﾝｶﾝ      (1=ﾐｷﾞ,2=ｳｴ,3=ﾋﾀﾞﾘ,4=ｼﾀ)
1290 '                  :    =5-8  ｷｰｽｷｬﾝ 1 ﾎｳｺｳ (5=ﾐｷﾞ,6=ｳｴ,7=ﾋﾀﾞﾘ,8=ｼﾀ)
1300 '                  :    =9,10 ｷｰｽｷｬﾝ 2 ﾎｳｺｳ (9=ｻﾕｳ,10=ｼﾞｮｳｹﾞ)
1310 '                  :    =99   ｼﾝﾛ ｶｳﾝﾀｰ=0 ﾍ
1320 '                  :                 [ﾂｷﾞ ﾊ ﾚｯﾄﾞｶｰ ﾉ ﾊﾞｱｲ]
1330 '                  :    =1-4  ﾎｳｺｳ ﾃﾝｶﾝ      (1=ｳｴ,2=ﾋﾀﾞﾘ,3=ｼﾀ,4=ﾐｷﾞ)
1340 'XADD(X),YADD(Y)   :ﾍﾝｲ X,Y
1350 'DOT(ﾜｸ,ｷｮﾘ)       :0=" ",1="･"
1360 'TRACE$(1)         :0=" ",1="･"
1370 'HISCR$            :HIGH SCORE
1380 'SCR$              :SCORE
1390 '
```

```
1400 '▬▬ MAIN ROUTINE ▬▬
1410 '―― COLD START ――
1420 DEFINT A-Z'                                    SET INTEGER
1430 HISCR=0
1440 DIM CRS(3,49),ADD(4,4),DOT(3,49)
1450   RESTORE 4130'                                READ CRS(3,49)
1460   FOR I=0 TO 3
1470     FOR J=0 TO 49
1480       READ CRS(I,J)
1490   NEXT J,I
1500   FOR I=1 TO 4'                                READ XADD(4),YADD(4)
1510     READ XADD(I),YADD(I)
1520   NEXT
1530   TRACE$(0)=" ":TRACE$(1)="・"'                SET TRACE$
1540 '
1550 '―― HOT  START ――
1560 GOSUB 1710'                                    GAME PLAY ?
1570 IF GAME=3 THEN 1670'                           PROGRAM END ?
1580 LEFT=3:SCR=0
1590   IF GAME>1 THEN GOSUB 1710:GOSUB 1830:GOTO 1570'  GAME OVER
1600   GOSUB 1870'                                  RESET CARS INITIAL POSITI
1610     IF GAME>0 THEN 1590'                       GAME BREAK
1620       IF MGO=1 THEN GOSUB 2650 ELSE MGO=MGO-1'  MOVE MYCAR OR STOP
1630       IF RGO=1 THEN GOSUB 2780 ELSE RGO=RGO-1'  MOVE REDCAR OR STOP
1640     GOTO 1610
1650 '
1660 '―― END ROUTINE ――
1670 END
1680 '
1690 '▬▬ SUB ROUTINE ▬▬
1700 '―― GAME REPLAY ――
1710 WIDTH 40,25:CONSOLE 0,25,0,1:COLOR 5,32,0:PRINT CHR$(12):COLOR 7
1720 LOCATE 2,10:PRINT "HI-SCORE:";HISCR
1730 LOCATE 2,12:PRINT "   SCORE:";SCR
1740 LOCATE 2,14:PRINT "      1――――――GAME START"
1750 LOCATE 2,15:PRINT "      2――――――GAME END"
1760 LOCATE 2,16:PRINT "                ・・・・・・・・・・HOW ??";
1770 I$=INPUT$(1):I=VAL(I$)
1780   IF I=1 THEN GAME=1:RETURN'                   GAME START
1790   IF I=2 THEN GAME=3:RETURN'                   GAME END
1800 GOTO 1760
1810 '
1820 '―― GAME OVER ――
1830 IF SCR>HISCR THEN HISCR=SCR'                   GET HIGH-SCORE ?
1840 RETURN
1850 '
1860 '―― SET CAR START POSITION ――
1870 GAME=0:DOT=129
1880 GOSUB 2020'                                    ｶﾞﾒﾝ
1890 RESTORE 4370'                                  SET DOT
1900 FOR I=0 TO 3
1910   FOR J=0 TO 49
1920     READ DOT(I,J)
1930 NEXT J,I
1940 XMYCAR=29:YMYCAR=22:C1=0:C2=0 :MCRS=2'         RESET MYCAR
1950 XRED=1   :YRED=22  :C3=0:C4=35:RCRS=2'         RESET REDCAR
1960 MGO=1:RGO=INT(RND(1)*10)+5'                    GO FLAG SET
1970   COLOR 7:LOCATE XMYCAR,YMYCAR:PRINT "◆";
1980   COLOR 2:LOCATE XRED,YRED:PRINT "●";
1990 RETURN
2000 '
2010 '―― ｶﾞﾒﾝ ――
2020 COLOR 5:PRINT CHR$(12);'                       PRINT
2030 FOR Y=1 TO 23
2040   X=(Y MOD 2)+1
2050   FOR X=X TO X+28 STEP 2
2060     LOCATE X,Y:PRINT "・";
2070 NEXT X,Y
2080 FOR I=0 TO 8 STEP 2'                           PRINT RECTANGLE
2090   IF (I MOD 8)=0 THEN COLOR 1 ELSE COLOR 5
2100   LOCATE I,I:PRINT "┌";
2110   Y1=I:Y2=24-I
2120   FOR X=I+1 TO 29-I
2130     LOCATE X,Y1:PRINT "─";:LOCATE X,Y2:PRINT "─";
2140   NEXT
2150   LOCATE X,Y1:PRINT "┐";:LOCATE X,Y2:PRINT "┘";
2160   X1=I:X2=30-I
2170   FOR Y=I+1 TO 23-I
```

```
2180      LOCATE X1,Y:PRINT "|";:LOCATE X2,Y:PRINT "|";
2190    NEXT
2200    LOCATE X1,Y:PRINT "└";
2210 NEXT
2220 X1=9 :X2=21:Y1=9 :Y2=15:GOSUB 2350'          CROSS OUT
2230 X1=14:X2=16:Y1=1 :Y2=23:GOSUB 2350
2240 X1=1 :X2=29:Y1=11:Y2=13:GOSUB 2350
2250 RESTORE 3860'                                PRINT TITLE
2260 FOR Y=1 TO 24
2270    READ I$:COLOR 2:LOCATE 31,Y:PRINT "◣";:COLOR 6:PRINT I$;
2280    COLOR 2:PRINT "◢";
2290 NEXT
2300 COLOR 2:LOCATE 10,10:PRINT "LEFT:    ";:GOSUB 2420 'PRINT MESSAGE
2310 COLOR 4:LOCATE 13,12:PRINT    "SCORE";:GOSUB 2490
2320 COLOR 4:LOCATE 10,14:PRINT "HI-SCORE";:GOSUB 2550
2330 RETURN
2340 '
2350 '—— CLEAR RECTANGLE ——————
2360 ' ——— PARA IN:X1,X2,Y1,Y2 —
2370 FOR Y=Y1 TO Y2
2380    FOR X=X1 TO X2
2390      LOCATE X,Y:PRINT " ";
2400 NEXT X,Y:RETURN
2410 '
2420 '—— PRINT LEFT ——
2430 LOCATE 15,10:PRINT "    ";'                  ERASE LAST MESSAGE
2440 IF LEFT<=1 THEN RETURN'                      LEFT=0 THEN ﾋｮｳｼﾞ ｾｽﾞ
2450 FOR X=15 TO 14+LEFT-1
2460    COLOR 7:LOCATE X,10:PRINT "◆";
2470 NEXT:RETURN
2480 '
2490 '—— PRINT SCORE ——
2500 SCR$=RIGHT$("0000"+RIGHT$(STR$(SCR),LEN(STR$(SCR))-1),5)
2510 COLOR 7:LOCATE 15,13:PRINT SCR$;
2520 DOT=DOT-1:IF DOT=0 THEN GAME=0'              DOT=0 ?
2530 RETURN
2540 '
2550 '—— PRINT HI-SCORE ——
2560 HISCR$=RIGHT$("0000"+RIGHT$(STR$(HISCR),LEN(STR$(HISCR))-1),5)
2570 COLOR 7:LOCATE 15,15:PRINT HISCR$;:RETURN
2580 '
2590 '—— CHANGE DIRECTION MYCAR ——
2600 XMYCAR=XMYCAR-1:YMYCAR=YMYCAR+1:MCRS=1:RETURN'     FOR RIGHT
2610 XMYCAR=XMYCAR+1:YMYCAR=YMYCAR+1:MCRS=2:RETURN'     FOR UP
2620 XMYCAR=XMYCAR+1:YMYCAR=YMYCAR-1:MCRS=3:RETURN'     FOR LEFT
2630 XMYCAR=XMYCAR-1:YMYCAR=YMYCAR-1:MCRS=4:RETURN'     FOR DOWN
2640 '
2650 '—— MOVE MYCAR ——
2660 COLOR 5:LOCATE XMYCAR,YMYCAR:PRINT " ";'     ERASE MYCAR
2670    CMEMO=CRS(C1,C2)'                         CMEMO=ｼﾝﾛ
2680    IF CMEMO=99 THEN C2=C2+1:GOTO 2670'       SKIP 99
2690    IF DOT(C1,C2)=1 THEN BEEP 1:SCR=SCR+10:GOSUB 2500:BEEP 0'   SCORE COUNT UP
2700    DOT(C1,C2)=0'                             ERASE DOT
2710    C2=C2+1:IF C2=50 THEN C2=0'               1-ROUND END ?
2720    ON CMEMO GOSUB 2600,2610,2620,2630,3030,3090,3150,3210,3270,3330
2730    XMYCAR=XMYCAR+XADD(MCRS):YMYCAR=YMYCAR+YADD(MCRS)
2740 COLOR 7:LOCATE XMYCAR,YMYCAR:PRINT "◆";'     PRINT MYCAR
2750 IF XRED=XMYCAR AND YRED=YMYCAR THEN 2910
2760 RETURN
2770 '
2780 '—— MOVE REDCAR ——
2790 COLOR 5:LOCATE XRED,YRED:PRINT TRACE$(DOT(C3,C4));'  ERASE REDCAR
2800    MEMO=CRS(C3,C4)'                          MEMO=ｷｭｳ ｼﾝﾛ
2810    C4=C4-1:IF C4<0 THEN C4=49'               ROUND END ?
2820    CMEMO=CRS(C3,C4)'                         CMEMO=ｼﾝﾛ
2830    IF CMEMO=99 THEN C4=C4-1:GOTO 2820'       SKIP 99
2840    ON CMEMO GOSUB 2980,2970,3000,2990,3390,3450,3510,3570,3630,3690
2850    XRED=XRED+XADD(RCRS):YRED=YRED+YADD(RCRS)
2860 COLOR 2:LOCATE XRED,YRED:PRINT "●";'         PRINT REDCAR
2870 IF XRED=XMYCAR AND YRED=YMYCAR THEN 2910
2880 RETURN
2890 '
2900 '—— CRASH MYCAR ——
2910 BEEP
2920 LEFT=LEFT-1:GOSUB 2430'                      PRINT LEFT OF MYCAR
2930 IF LEFT=0 THEN GAME=2 ELSE GAME=1'           GAME OVER PRINT
2940 RETURN
2950
```

```
2960 '—— CHANGE DIRECTION REDCAR ——
2970 XRED=XRED+1:YRED=YRED+1:RCRS=3:RETURN'      FOR LEFT
2980 XRED=XRED-1:YRED=YRED+1:RCRS=2:RETURN'      FOR UP
2990 XRED=XRED-1:YRED=YRED-1:RCRS=1:RETURN'      FOR RIGHT
3000 XRED=XRED+1:YRED=YRED-1:RCRS=4:RETURN'      FOR DOWN
3010 '
3020 '—— GO RIGHT? MYCAR ——
3030 GOSUB 3740:IF KY$<>"R" THEN RETURN'         KEY OFF
3040   XMYCAR=XMYCAR+2'                          CHANGE LOCATE
3050   IF MCRS=2 THEN C1=C1-1 ELSE C1=C1+1'             CORSE
3060 RETURN
3070 '
3080 '—— GO UP? MYCAR ——
3090 GOSUB 3740:IF KY$<>"U" THEN RETURN'         KEY OFF
3100   YMYCAR=YMYCAR-2'                          CHANGE LOCATE
3110   IF MCRS=1 THEN C1=C1+1 ELSE C1=C1-1'             CORSE
3120 RETURN
3130 '
3140 '—— GO LEFT? MYCAR ——
3150 GOSUB 3740:IF KY$<>"L" THEN RETURN'         KEY OFF
3160   XMYCAR=XMYCAR-2'                          CHANGE LOCATE
3170   IF MCRS=2 THEN C1=C1+1 ELSE C1=C1-1'             CORSE
3180 RETURN
3190 '
3200 '—— GO DOWN? MYCAR ——
3210 GOSUB 3740:IF KY$<>"D" THEN RETURN'         KEY OFF
3220   YMYCAR=YMYCAR+2'                          CHANGE LOCATE
3230   IF MCRS=1 THEN C1=C1-1 ELSE C1=C1+1'             CORSE
3240 RETURN
3250 '
3260 '—— GO LEFT OR RIGHT? MYCAR ——
3270 GOSUB 3740'                                 KEYSCAN
3280   IF KY$="L" THEN 3160'                     LEFT
3290   IF KY$="R" THEN 3040'                     RIGHT
3300 RETURN
3310 '
3320 '—— GO UP OR DOWN? MYCAR ——
3330 GOSUB 3740'                                 KEYSCAN
3340   IF KY$="D" THEN 3220'                     LEFT
3350   IF KY$="U" THEN 3100'                     RIGHT
3360 RETURN
3370 '
3380 '—— GO RIGHT? REDCAR ——
3390 IF C3=C1 OR MEMO<>0 THEN RETURN
3400   XRED=XRED+2'                              CHANGE LOCATE
3410   IF RCRS=2 THEN C3=C3+1 ELSE C3=C3-1'             CORSE
3420 RETURN
3430 '
3440 '—— GO UP? REDCAR ——
3450 IF C3=C1 OR MEMO<>0 THEN RETURN
3460   YRED=YRED-2'                              CHANGE LOCATE
3470   IF RCRS=1 THEN C3=C3-1 ELSE C3=C3+1'             CORSE
3480 RETURN
3490 '
3500 '—— GO LEFT? REDCAR ——
3510 IF C3=C1 OR MEMO<>0 THEN RETURN
3520   XRED=XRED-2'                              CHANGE LOCATE
3530   IF RCRS=2 THEN C3=C3-1 ELSE C3=C3+1'             CORSE
3540 RETURN
3550 '
3560 '—— GO DOWN? REDCAR ——
3570 IF C3=C1 OR MEMO<>0 THEN RETURN
3580   YRED=YRED+2'                              CHANGE LOCATE
3590   IF RCRS=1 THEN C3=C3+1 ELSE C3=C3-1'             CORSE
3600 RETURN
3610 '
3620 '—— GO LEFT OR RIGHT? REDCAR ——
3630 IF C3=C1 OR MEMO<>0 THEN RETURN
3640   IF C3>C1 THEN C3=C3-1:IF RCRS=2 THEN XRED=XRED-2  ELSE XRED=XRED+2
3650   IF C3<C1 THEN C3=C3+1:IF RCRS=2 THEN XRED=XRED+2  ELSE XRED=XRED-2
3660 RETURN
3670 '
3680 '—— GO UP OR DOWN? REDCAR ——
3690 IF C3=C1 OR MEMO<>0 THEN RETURN
3700   IF C3>C1 THEN C3=C3-1:IF RCRS=1 THEN YRED=YRED-2  ELSE YRED=YRED+2
3710   IF C3<C1 THEN C3=C3+1:IF RCRS=1 THEN YRED=YRED+2  ELSE YRED=YRED-2
3720 RETURN
3730 '
```

```
3740 '—— KEYSCAN ——
3750 I0=INP(0):I1=INP(1):KY$=""'                    GET CHECK KEY ON
3760 IF I0<>255 OR I1<>255 THEN 3780'              KEY ON
3770   IF RND(1)*100<95 THEN RETURN ELSE MGO=INT(RND(1)*10)+1:RETURN'   STOP
3780   IF I0=239 THEN KY$="L":RETURN'              LEFT ON
3790   IF I0=191 THEN KY$="R":RETURN'              RIGHT ON
3800   IF I0=251 THEN KY$="D":RETURN'              DOWN ON
3810   IF I1=254 THEN KY$="U":RETURN'              UP ON
3820 RETURN
3830 '
3840 '▬▬▬ DATA AREA ▬▬▬
3850 '—— TITLE DATA ——
3860  DATA "■▔▔▔◣"
3870  DATA "■▁▁▁◤"
3880  DATA "■   ◣"
3890  DATA "■▁▁▁◤"
3900  DATA "◢▔▔▔◣"
3910  DATA "■   ■"
3920  DATA "■   ■"
3930  DATA "◥▁▁▁◤"
3940  DATA "◣   ◢"
3950  DATA "■◣▁◢■"
3960  DATA "■ ▔ ■"
3970  DATA "■   ■"
3980  DATA "■▔▔▔◣"
3990  DATA "■▁▁▁◤"
4000  DATA "■   ◣"
4010  DATA "■▁▁▁◤"
4020  DATA "■▔▔▔◥"
4030  DATA "■▁▁  "
4040  DATA "■    "
4050  DATA "■▁▁▁◢"
4060  DATA "■▔▔▔◣"
4070  DATA "■▁▁▁◤"
4080  DATA "■ ◥◣ "
4090  DATA "■▁ ◥◣"
4110 '
4120 '—— COURSE DATA ——
4130 DATA 00,00,00,00,07,07,00,00,00,00,03:'                COURSE-0
4140 DATA 00,00,00,00,00,08,08,08,00,00,00,00,00,04
4150 DATA 00,00,00,00,05,05,00,00,00,00,01
4160 DATA 00,00,00,00,00,06,06,06,00,00,00,00,00,02
4170 DATA 00,00,00,99,09,09,00,00,00,99,03:'                COURSE-1
4180 DATA 00,00,00,00,99,10,10,10,00,00,00,00,99,04
4190 DATA 00,00,00,99,09,09,00,00,00,99,01
4200 DATA 00,00,00,00,99,10,10,10,00,00,00,00,99,02
4210 DATA 00,00,99,99,09,09,00,00,99,99,03:'                COURSE-2
4220 DATA 00,00,00,99,99,10,10,10,00,00,00,99,99,04
4230 DATA 00,00,99,99,09,09,00,00,99,99,01
4240 DATA 00,00,00,99,99,10,10,10,00,00,00,99,99,02
4250 DATA 00,99,99,99,05,05,00,99,99,99,03:'                COURSE-3
4260 DATA 00,00,99,99,99,06,06,06,00,99,99,99,04
4270 DATA 00,99,99,99,07,07,00,99,99,99,01
4280 DATA 00,00,99,99,99,08,08,08,00,00,99,99,99,02
4290 '
4300 '—— ADD DATA ——
4310 DATA +2, 0:'                                  RIGHT
4320 DATA  0,-2:'                                  UP
4330 DATA -2, 0:'                                  LEFT
4340 DATA  0,+2:'                                  DOWN
4350 '
4360 '—— INITIAL DOT DATA ——
4370 DATA 1,1,1,1,1,0,1,1,1,1,1:'                  COURSE-0
4380 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
4390 DATA 1,1,1,1,1,0,1,1,1,1,1
4400 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
4410 DATA 1,1,1,1,1,0,1,1,1,1,1:'                  COURSE-1
4420 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
4430 DATA 1,1,1,1,1,0,1,1,1,1,1
4440 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
4450 DATA 1,1,1,1,1,0,1,1,1,1,1:'                  COURSE-2
4460 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
4470 DATA 1,1,1,1,1,0,1,1,1,1,1
4480 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
4490 DATA 1,1,1,1,1,0,1,1,1,1,1:'                  COURSE-3
4500 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
4510 DATA 1,1,1,1,1,0,1,1,1,1,1
4520 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
```

リアルタイムゲーム「BOMBER」挑戦
バリエーション&マシン語

はじめに

〈内　用　薬〉

薬品名：GAMINGへの招待

用　法：1か月1回

　　　　夕食後の30分とねる前

効　用：GAME作りに嫌気がさす

注）1．服用後，頭が混乱し，バグの山に悩まされることがあります。そのときは，使用をただちに中断してください。

　　2．服用にあたっては，使用上の注意を良く読み，正しく用いましょう。

（PIPSエレキバン薬品）

デラックス版の完成

まず第6章との変更点です。

・どこが変わった？

――まずGAMEが，面白くなりました。

バリエーションの追加

をすることにより，BASICのGAMEとしては，わりと（かなり？）面白くすることができました。

・LISTが長くなった？

――ハイ。第6章までのリストに比べてかなり長くなりました。これは，せっかくGAMEを仕上げたのだから最後の仕上げとして

デモンストレーション

を追加し，**マシン語を取り入れた**ためです。

・LISTのおしまいの方に追加されたゴチャゴチャしたDATAは何？

これは，PC－8001のユーザーに対しての私からの

アフター・サービス

です。ゴチャゴチャしたDATAは，実はマシン語のDATAです。ということは，――。

そうです。第7章で紹介するPC－8001用の参考プログラムは，

〝BASIC＋マシン語版〞

になっているのです。マシン語の部分にはGAMEの中に

音楽

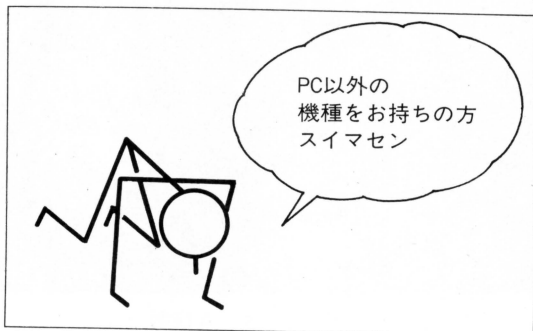
電子音

を取り入れるためのルーチンが記述されています。

音に弱いPC－8001

のための苦肉の策ですね。ですからPCのユーザーだけは，このプログラムを入力していただくだけで，

〝BOMBER〞

を楽しむことができます（他のマシンのユーザーの皆さんゴメンなさい）。

新ルール

プログラムをRUNすると，

WAIT A MOMENT！

のメッセージが出て，1～2秒点滅します（**写真19**）。この間に，

マシン語のメモリへの書き込み

変数の初期化

配列変数へのDATAの読み込み

等が行なわれます。

準備が終わると，**MUSIC**が流れます。タイトルは，

We Shall Over Come！

「GAMEに打ち勝つまでは，——」

というわけです。この音楽，どこかで聞いたことがありませんか？　そうなんです，××さん。

〝インデアン・ポーカー〟

（81年7月号発表）

で用いたのと同じです。これは塚越氏の幅狭い音楽感覚のなせるワザで，他に適当なオープニング・ミュージックが見つからなかったからです。

続いて，

耳をつんざくような電子音

が聞こえ，GAMEのインストラクションが表示されます（**写真20**）。

このモードで選択できるのは，

キー[1]——GAME START

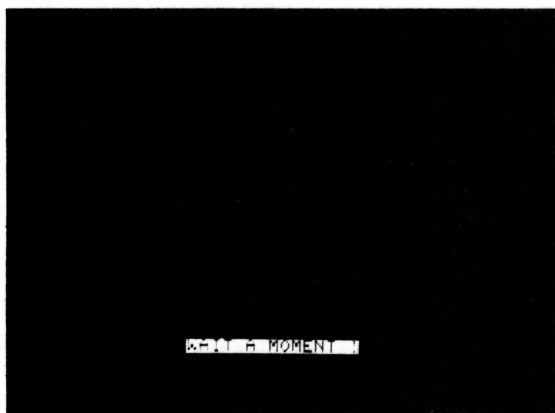
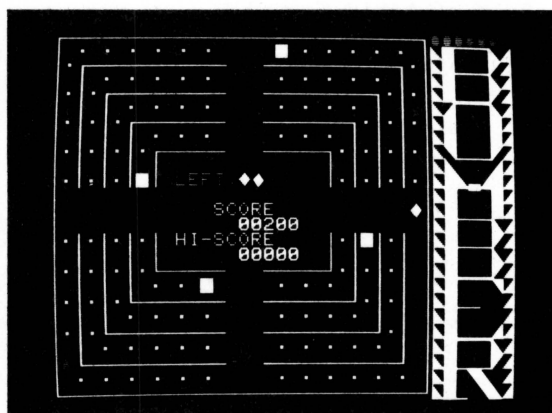
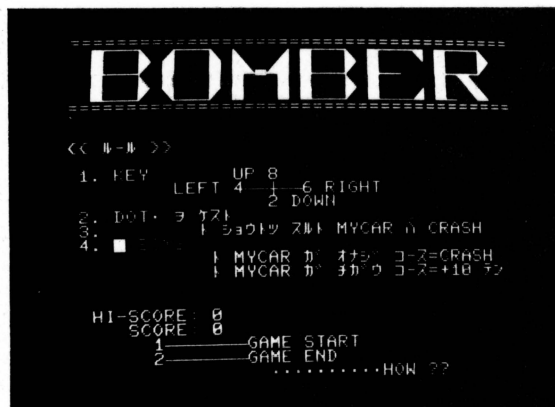
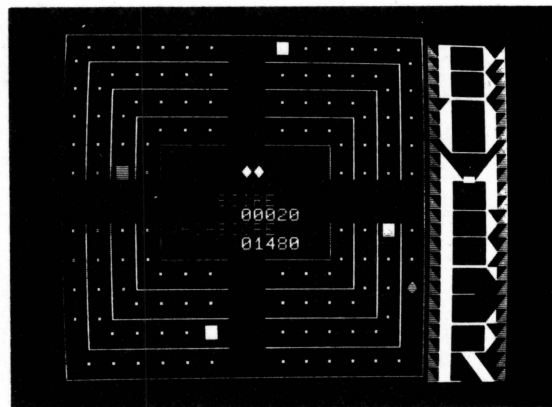
キー[2]——GAME END

で前章と同じです。

ここで[2]のキーを押して，アッ，イャ[2]のキーを押すとGAMEが終わってしまいますから，やはり[1]のキーを押しましょう。

[1]のキーを押すと，**ファンファーレ**が鳴り，画面はGAMEエリアに一変します（**写真21**）。

タイトルの上方を見ると，

と赤いものが並んでいます。これは一体何でしょうね？　もちろん**GAMEの新ルール**に関係があります。

《**写真19**》 **RUN直後の表示**

《**写真21**》**BOMBER登場**

《**写真20**》 **ゲーム説明**

《写真4》 むむっ！ これは何だ

まだ画面には，MYCAR，REDCARが現われていません。

しばらくすると，四つの

■……BOMB

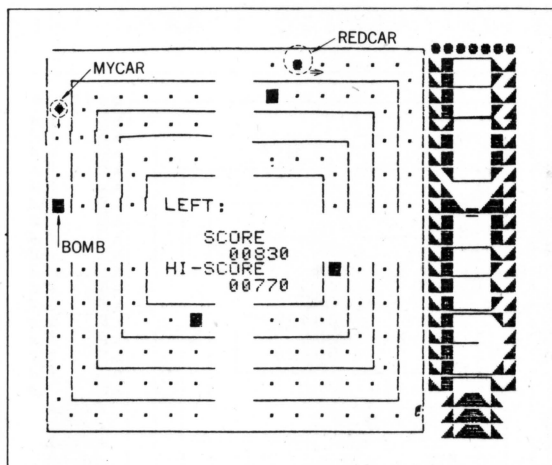
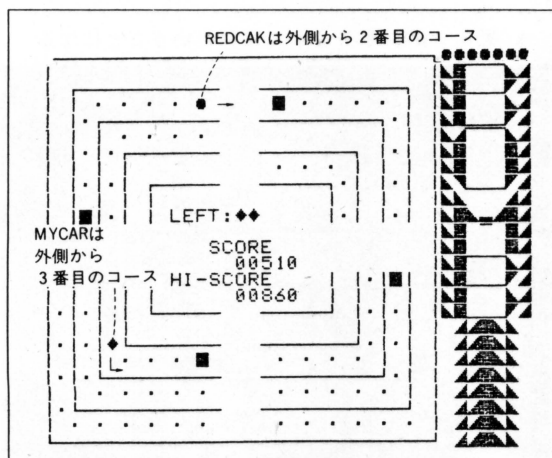
が現われ（これも新ルールです），MYCAR◆が動き出します（**写真22**）。まずここで**ドット・を消す時の音が変わった**のに気付くと思います。BEEP音では面白くありませんから，**電子音**に変えておきました。

それではここで新ルールを説明しておくことにします。

① **BOMB■**

この**BOMBの登場**によってこのGAME，かなり難しくなっています。BOMBは，最初各コーナーの出口に一つずつ，**計四つが乱数で配置**されます。このBOMBは，二つの役割りを持ちます。

1) **MYCARとREDCARが同一コース上にあるとき**（**第3－59図**）

このとき，BOMBは

《第3-59図》MYCARとREDCARが同一コースにある

BOMB＝**爆弾**

として働きます。したがってMYCARが，BOMBに衝突すると，爆発を起こし，

MYCARは破壊

されます。

2) **MYCARとREDCARが異なるコースにいるとき**（第3－60図）。

このときは，

BOMB■＝DOT・

として働きます。ですからMYCARが，BOMBの上を通れば，

10点が加算

されます。

〔解　説〕

GAMEで高得点を得るには，**いかにタイミングよくBOMB■を消すか**にかかっています。たとえば**第3－61図**のようにMYCARとREDCARが異なるコースにあるからといって，安心してBOMB■のあるコースに入ると（**第3－62図**），やがて**REDCARはコースを変更**し，

MYCARのコース
＝REDCARのコース

ということになります。もうこうなると，

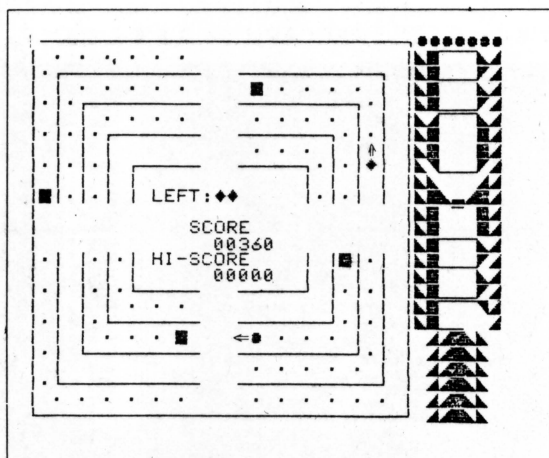
CRASH!

を避けることはできません（**第3－63図**）。

したがってこのGAMEでは，できるだけ早く**先を読み**，かつ**高度(?)の反射神経を駆使**することが要求されます。

② **HEAVY－BOMB●●●●●●●**

新ルールは，それだけではありません。BOMBに気を取られ，グズグズしていると

《第3-60図》MYCARとREDCARが異なるコースにある

《第3-61図》コースが異なると思って安心していると

HEAVY－BOMB

が大爆発を起こし，たちまち

GAME　OVER！

となってしまいます。

HEAVY－BOMBは，画面の右側，タイトルの上方にあります。そしてこの**タイトル**がくせ者で，実は

タイトル＝導火線

となっています。GAME　START直後，導火線に火がつけられます（**写真23**）そして一刻一刻と爆発が近づきます（**写真24**）。導火線が伸びるのは，伸びる毎に

イヤラシイ電子音

がしますから，すぐに分かります。

このようにHEAVY－BOMBの存在があるため，**できるだけ効率良くDOT・を消していく**必要があります。REDCARやBOMBを避けるためむやみにDOT・のないコースを選んだりしていると，だんだん時間がなくなり，

HEAVY－BOMBの大爆発

ということになります。HEAVY－BOMBの爆発をくい止める方法はただ一つ，**爆発前にすべてのDOT・を消してしまうこと**です。そうすれば，

一面の終了

となります。

GAME OVER

新ルールの導入は，以上の通りです。もちろん

MYCARの故障etc.

のルールは，前章の完成品と同じです。ですからゲームは，

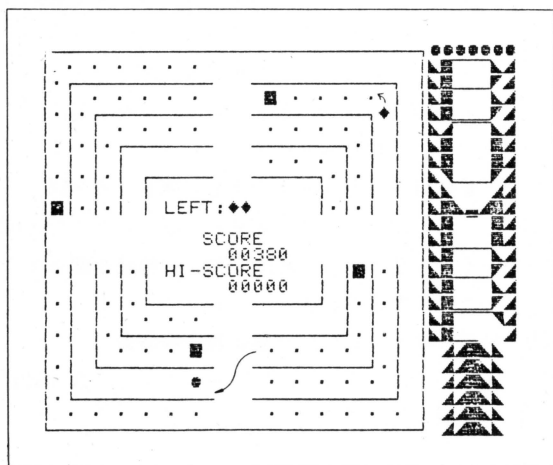
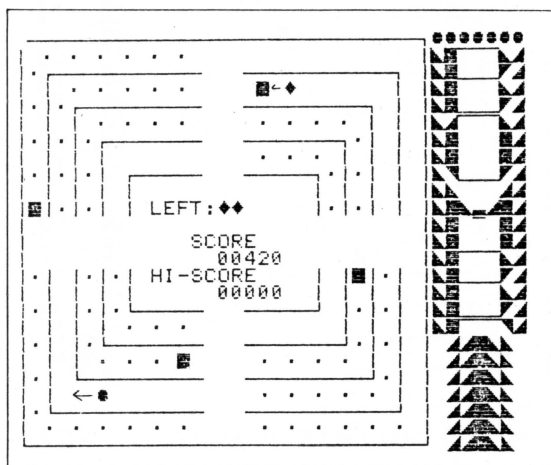
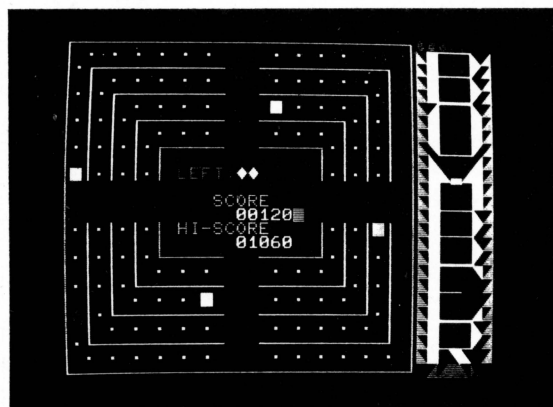
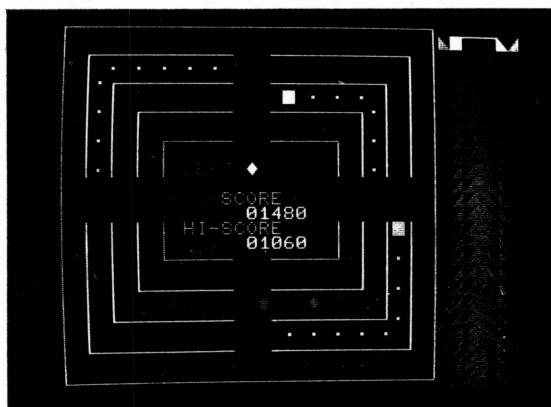
オリジナル・ルール

前章までの**バリエーション**

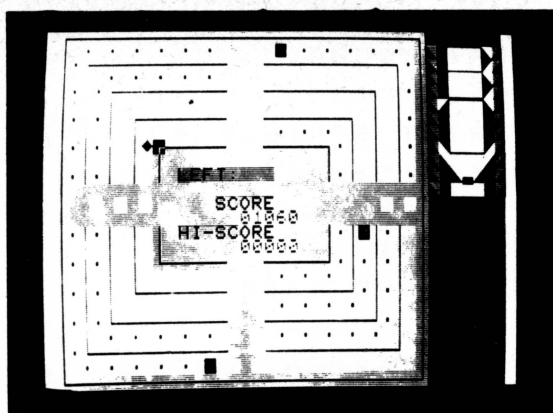
追加の**新ルール**

をおりまぜながら進行していくことになります。**写真25**は，MYCARとREDCARが，

CRASH！

して画面が反転したところ，また**写真26**はGAMEが

《第３－６２図》　REDCARはコースを変えてきて……

《第３－６３図》　やがて爆発ということになる

《写真23》　導火線に火がついたー

《写真24》　危な〜〜い　爆発だ！

《写真25》 CRASH！

《写真26》 これが最後のMYCAR

進み，MYCARが残り0となっています。こうして

① すべてのMYCARがなくなる

② HEAVY BOMBが大爆発

のいずれかが起こると，

GAME OVER

となります。そのとき**敗因の種類**が，メッセージとして表示されます（**写真27**）。その後，画面は**写真28**の状態に戻ります。なおここで表示されている得点は，今やったGAMEの得点，また最高点は今までのGAMEの最高点です。

以上が遊び方の説明です。

《写真27》 GAME OVER

《写真28》 ゲームの天才？

慶安の御触書ホイ！

慶安2年（1649年）に農民生活の規定として制定された**慶安の御触書の補遺**が，何と1982年10月，東京東五反田某新聞社敷地内で発見された（**第3－64図**）。破損の状態は，はなはだしいが，かろうじて次の如く解読された。

《第3－64図》 大発見！

1．朝ハ早オキヲシ，GAMEヅクリニハゲムコト。

2．「GAMEINGヘノ招待」ハ，継続シテ読ミ，カツヨク保管シテオクコト。

3．保管資料ガタマッタラ，スミヤカニ塵紙交換ニ出スコト。

4．第3ブロックノシリーズノLISTヲ読ムニアタッテ他機種ノ人ハ，次節ヲ参考ニスルコト。

5．第3ブロックノシリーズガ少シ難シカッタ人ハ，ワカルトコロ（前書キダケ？）ヲ読ンデオキ，後日理解ニ努メルコト。ソレデモダメナトキハ，アッサリアキラメルコト。
6．第3ブロックノシリーズガ易シスギタ人ハ，筆者ヲ馬鹿ニスルコト。

——以上

方言の補足

今章のリストはPC－8001上で開発してあり，そのBASICの方言等については，その都度補足してきました。しかしながら最終章の今，ここにまとめておきたいと思います。リストを読むときの参考にしてください。

① **CLEAR**
DEF USR1＝&HCC00
I＝USR1（0）
OUT 81，33

これらの命令は，**マシン語**をいじくるための命令または**IOポートにDATAを出力**する命令です。音楽等を出力するのに用いています。音の出せる機種の方は，自分のマシンに合わせて変更してください。

② **DEFINT A－Z**

変数を整数系に指定しています。リアルタイム・ゲームを作るときは，だいたい整数系の変数で間に合います。

③ **I$＝INPUT$（1）**

LEVEL3以上のBASICであれば，たぶんこの命令はあるでしょう。INPUT文で代用できます。キーボードより一文字入力し，I$に格納します。

④ **IF （I MOD 8）＝0 THEN～**

MODは，割算の余りを求めるのに使います。この文の意味は，

〝Iを8で割った余りが0ならば，THEN以下を実行しなさい〟

というものです。

⑤ **KEYSCANルーチン（4730～4810行）**

このルーチンについては，第4章で分析してみましたので，そちらを御覧になってください。

⑥ **LINE（5，5）－（33，19），"■"，2，BF**

画面上の任意の位置に任意の大きさの長方形を描き，任意の色でぬりつぶす命令です。

⑦ **PRINT CHR$（12）**

画面クリアの命令です。

⑧ **PRINT STRING$（39，"＝"）**

ある行を＝で埋めます。

PRINT 〝＝＝………＝〟

で代用できます。

⑨ **RESTORE 5580**

この命令は，たぶんあなたのマシンにもあるでしょう。DATA文を読み出す位置を指定します。

⑩ **WIDTH 40，25**
CONSOLE 0，25，0，0

この命令は，画面モードの設定を行うものです。概略を言えば，

｛画面サイズ＝40×25
｛画面モード＝カラー

にセットされます。

HEAVY BOMBのプログラミング

続いて追加した**〈新ルール〉**のプログラミングについて**分析**してみることにします。

最初に簡単な**HEAVY－BOMB**から。

HEAVY－BOMBの爆発管理は，

変数TBOM

によって行なっています。

TBOMの値は，最初は0です。これは，

「MYCAR，REDCAR
の位置をスタートにセット」

するルーチンでおこなっています（2650行）。

第6章で見ましたようにGAMEが進行中の間プログラムは，

メインルーチン：1940行～1980行

の間をグルグルまわっています。このループを1回まわるたびに（故障が起きていないかぎり），車は**1ドット**・ずつ進みます。そして1970行を見ればわかるように，TBOMの値も一つ増えます。すなわちループを1回まわる（＝車が1ドット進む）間

TBOM：0→1→2→3→…

と大きくなっていきます。

さて，このまま放っておきますとTBOMの値は，エラーになるまで大きくなっていきます。ところが1970行を良く見ますと，

TBOM＝7

になったとき，

TBOM＝0

に戻してやり，3680行からのサブルーチンをCallしているのがわかります。そこでこの**サブルーチン**の働きを調べてみることに致しましょう。

このルーチンは，**炎を伸ばしていく**ルーチンです（3670～3710行）。このルーチンに飛び込む毎に，

炎 ▲▲▲

が一つずつ伸びていきます。炎の長さの管理をしているのが，

変数ＢＯＭ

で初期値は，

ＢＯＭ＝１（2650行）

です。以上のことより，このルーチンの働きをまとめると，次のようになります。

① 炎を伸ばすときのサウンドを鳴らす（3680行）。
② 炎を描く（3690行）。
③ 変数ＢＯＭの値を一つ大きくする。もし

ＢＯＭ＝25

となると，炎がＨＥＡＶＹ－ＢＯＭＢに到達したことになりますから，

ＧＡＭＥ＝２（ＧＡＭＥ　ＯＶＥＲ）
ＣＡＵＳＥ＝２（敗因＝時間切れ）

をセットしてリターンします（3920～3930行）。

以上のようにＨＥＡＶＹ－ＢＯＭＢの管理は，

ＴＢＯＭ＝炎を伸ばすタイミング
ＢＯＭ＝爆発のタイミング

の**二つの変数**で行っているわけです。この様子をまとめたのが，**第３－６５図**です。

《第３－６５図》 HEAVY－BOMB処理

《第３－６６図》 いきなりこれでは避けようがない

ＢＯＭＢの処理

次が，**ＢＯＭＢ■の処理**です。

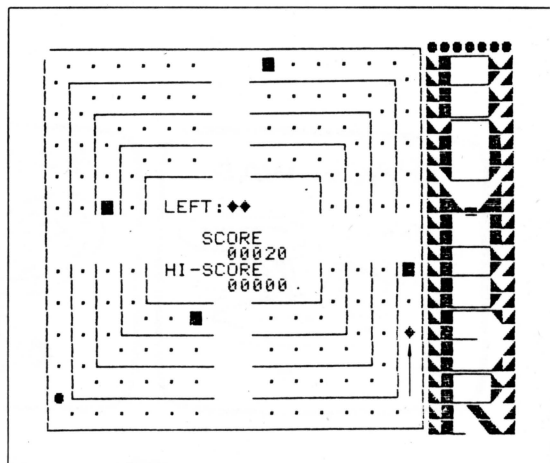
ＢＯＭＢの配置は，車類の初期化ルーチンの中で，行っています（2710行）。それは，2790行～2850行の〝ＳＥＴ　ＢＯＭＢ〞をＣＡＬＬすることで実現しています。

配置の仕方は，

① 各コーナーの最後にコースを乱数で決める。
② 最初のコーナーだけは，一番外側に置かないようにしています。さもないと，ＭＹＣＡＲが動き出したとたん，ＲＥＤＣＡＲと同一コースにいる関係上爆発を起こしてしまうからです（**第３－６６図**）。
③ 位置が決まったら道路状況を表わす

配列変数ＤＯＴ（コース，キョリ）＝２

にセットしてやります。

以上のように準備されたＢＯＭＢを，各ＣＡＲが通過した場合の処理を考えてみましょう。

まず**ＲＥＤＣＡＲ**から。

ＲＥＤＣＡＲが通過しても**何も起こりません**。3750行を見てください。もしＲＥＤＣＡＲが通過した位置がＢＯＭＢのときは，

ＰＲＩＮＴ
　ＴＲＡＣＥ＄（ＤＯＴ（Ｃ３，Ｃ４））
　　　　　　　　　‖
　　　　　　　　　２

のように～～の部分が２となります。それは，先程の

ＳＥＴ　ＢＯＭＢ（2810～2840行）

でそのようにセットしてあるからです。ですから

ＰＲＩＮＴ　ＴＲＡＣＥ＄（２）

ということになります。ところが，1720行で

ＴＲＡＣＥ＄（２）＝〝■〞

に定義してあります。したがってREDCARが，BOMBの上を通過しても何も起こりません。

次に**MYCARの場合**です。

MYCARが，BOMBの上を通過した場合，すなわち

DOT（C1，C2）＝2

のときは，3570行で処理しています。ここでは，

C1＝C3

すなわち，

MYCARのコース＝REDCARのコース

か否かで**異なる処理**を行っています。コースが異なるときは，

DOT（C1，C2）＝1

すなわち，**単にドット・上を通過したもの**と見なしていますが，同一コース上にあるときは，爆発させています。それが，

CRASH　MYCAR（3870〜3930行）

です。

以上のしくみを見てもおわかりのように，プログラムを走らせてみると難しそうに見えることでも，実際は単純な

変数の処理

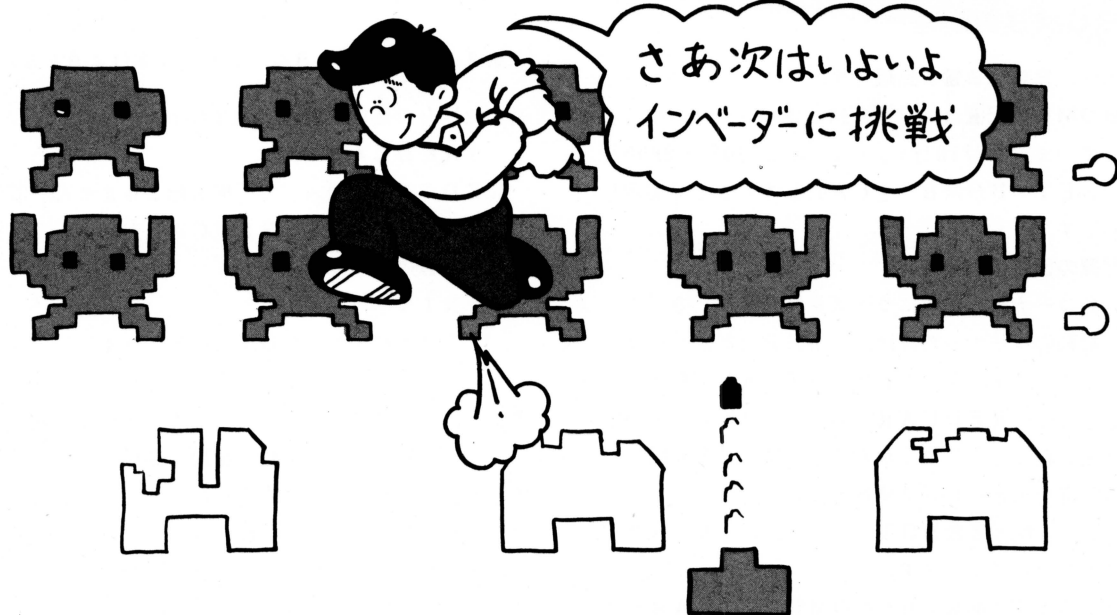
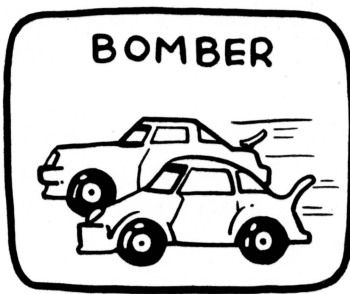
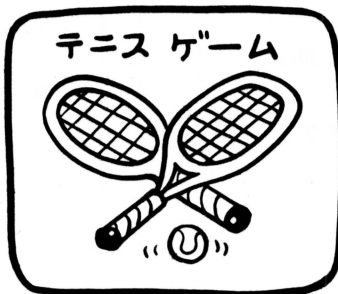
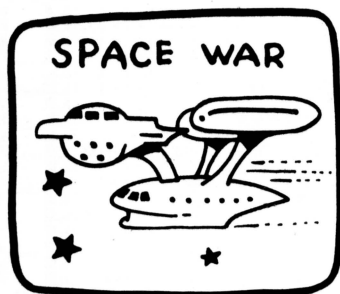
を行っているだけのことだったのです。おわかりいただけたでしょうか？

GAMINGへの招待

以上7章にわたって作り続けてきた

"BOMBER"

も今ここにようやく完成し，ホッとしているところです。ちょっと見ると難しそうなゲームも，一つ一つ分解していくと，わりと簡単なプログラムの集まりなんですね。ここで紹介したリアルタイムゲームの作り方を参考にあなた独自のオリジナルゲームの開発に役立てて下さい。

それでは，いよいよ第4ブロックではインベーダー・ゲームに挑戦してみましょう！

```
1000 '♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
1010 '♥  BOMBER for 'INVITATION FOR GAMING'2-7   ♥
1020 '♥                << 82.2.15-9.1 >>        ♥
1030 '♥                      by K.TUKAGOSHI ♥
1040 '♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
1050 '
1060 '——'VARIABLE ——
1070 'HISCR              :ﾊｲ ｽｺｱ
1080 'SCR                :ｽｺｱ
1090 'LEFT               :COUNTER OF LEFT CAR
1100 'XMYCAR,YMYCAR      :LOCATE MYCAR
1110 'C1,C2              :MYCAR ｺｰｽ & ｷｼﾞｭﾝﾃﾝ ｶﾗ ﾉ ｷｮﾘ
1120 'MCRS               :DIRECTION OF MYCAR      (1=ﾐｷﾞ,2=ｳｴ,3=ﾋﾀﾞﾘ,4=ｼﾀ)
1130 'CMEMO              :MEMO of MCRS(C1,C2) OR RCRS(C3,C4)
1140 'MEMO               :MEMO of OLD RCRS(C3,C4)
1150 'XRED,YRED          :LOCATE REDCAR
1160 'C3,C4              :REDCAR ｺｰｽ & ｷｼﾞｭﾝﾃﾝ ｶﾗ ﾉ ｷｮﾘ
1170 'RCRS               :DIRECTION OF REDCAR
1180 'I0,I1              :VALUE OF INP
1190 'MGO                :MYCAR START FLAG (1=OK,1<>NO)
1200 'RGO                :REDCAR START FLAG (1=OK,1<>NO)
1210 'DOT                :LEFT OF DOT
1220 'GAME               :GAME FLAG (0=GAME ON,1=CRASH,2=GAME OVER)
1230 'BOM                :COUNTER OF LEFT TIME
1240 'TBOM               :TIMER OF BOMB
1250 'CAUSE              :CAUSE OF GAME OVER (1=MYCAR,2=TIMER)
1260 '
1270 '—— DIMENSION ——
1280 'CRS(ｺｰｽ,ｷｮﾘ)       :ｺｰｽ=0,1,2,3 (0=OUTSIDE,3=INSIDE)
1290 '                   :ｷｮﾘ=0    ｾﾞﾝｼﾝ  [ﾂｷﾞ ﾊ ﾏｲｶｰ ﾉ ﾊﾞｱｲ]
1300 '                   :   =1-4  ﾎｳｺｳ ﾃﾝｶﾝ      (1=ﾐｷﾞ,2=ｳｴ,3=ﾋﾀﾞﾘ,4=ｼﾀ)
1310 '                   :   =5-8  ｷｰｽｷｬﾝ 1 ﾎｳｺｳ (5=ﾐｷﾞ,6=ｳｴ,7=ﾋﾀﾞﾘ,8=ｼﾀ)
1320 '                   :   =9,10 ｷｰｽｷｬﾝ 2 ﾎｳｺｳ (9=ｻﾕｳ,10=ｼﾞｮｳｹﾞ)
1330 '                   :   =99   ｼﾝﾛ ｶｳﾝﾀｰ=0 ﾍ
1340 '                   :              [ﾂｷﾞ ﾊ ﾚｯﾄﾞｶｰ ﾉ ﾊﾞｱｲ]
1350 '                   :   =1-4  ﾎｳｺｳ ﾃﾝｶﾝ      (1=ｳｴ,2=ﾋﾀﾞﾘ,3=ｼﾀ,4=ﾐｷﾞ)
1360 'XADD(X),YADD(Y)   :ﾍﾝｲ X,Y
1370 'DOT(ｺｰｽ ｷｮﾘ)       :DOT CONDITION (0=" ",1="・",2="■")
1380 'TRACE$(1)          :FOR ERASE REDCAR (0=" ",1="・",2="■")
1390 'GM$(3),0V$(3)      :DATA OF GAME OVER
1400 'CAUSE$(2)          :DATA OF CAUSE
1410 'TITLE$(3)          :DATA OF TITLE
1420 'HISCR$             :HIGH SCORE
1430 'SCR$               :SCORE
1440 'KY$                :VALUE OF KEYSCAN (KIY OFF="")
1450 '
1460 '▬▬ MAIN ROUTINE ▬▬
1470 '—— SYSTEN INITIAL ——
1480 CLEAR 300,&HCBFF:WIDTH 40,25:CONSOLE 0,25,0,0
1490 COLOR 6:LOCATE 10,20:PRINT "WAIT A MOMENT !":COLOR 0
1500 RESTORE 5580
1510 FOR I=&HCC00 TO &HCDA0'                       WRITE MACHINE LANGUAGE
1520   READ J$:POKE I,VAL("&h"+J$)
1530 NEXT
1540 DEF USR1=&HCC00'                              ON PRG START
1550 DEF USR2=&HCC19'                              ON GAME START
1560 DEF USR3=&HCC2B'                              ON MYCAR CRASH
1570 DEF USR4=&HCC31'                              ON ERASE DOT
1580 DEF USR5=&HCC42'                              ON INCREASE TIMER
1590 '
1600 '—— COLD START ——
1610 DEFINT A-Z'                                   SET INTEGER
1620 HISCR=0
1630 DIM CRS(3,49),ADD(4,4),DOT(3,49)
1640 DIM TRACE$(2),GM$(3),OV$(3),CAUSE$(3),TITLE$(3)
1650   RESTORE 5160'                               READ CRS(3,49)
1660   FOR I=0 TO 3
1670     FOR J=0 TO 49
1680       READ CRS(I,J)
1690   NEXT J,I
1700   FOR I=1 TO 4'                               READ XADD(4),YADD(4)
1710     READ XADD(I),YADD(I)
1720   NEXT
1730   TRACE$(0)=" ":TRACE$(1)="・":TRACE$(2)="■"'SET TRACE$
1740   RESTORE 5100
1750   FOR I=0 TO 3'                               DATA OF GAME OVER
1760     READ GM$(I),OV$(I)
1770   NEXT
1780   RESTORE 4850:FOR I=0 TO 3:TITLE$(I)="":NEXT'  READ DATA OF TITLE
1790   FOR I=1 TO 6
1800     FOR J=0 TO 3
1810       READ J$:TITLE$(J)=TITLE$(J)+J$+" "
```

```
1820   NEXT J,I
1830   CAUSE$(1)=" ALL MYCAR IS CRASHED! "
1840   CAUSE$(2)="      TIME OUT !       "
1850 ' I=USR1(0)'                                   OPENNING MUSIC
1860 '
1870 '—— HOT  START ——
1880 GOSUB 2130'                                    GAME PLAY ?
1890 IF GAME=3 THEN 2010'                           PROGRAM END ?
1900 PRINT CHR$(12):I=USR2(0)'                      GAME START SOUND
1910 LEFT=3:SCR=0
1920   IF GAME>1 THEN GOSUB 2050:GOSUB 2130:GOSUB 2610:GOTO 1890'  GAME OVER
1930   GOSUB 2650'                                  RESET CARS INITIAL POSITION
1940     IF GAME>0 THEN 1920'                       GAME BREAK
1950       IF MGO=1 THEN GOSUB 3520 ELSE MGO=MGO-1'  MOVE MYCAR OR STOP
1960       IF RGO=1 THEN GOSUB 3730 ELSE RGO=RGO-1'  MOVE REDCAR OR STOP
1970       TBOM=TBOM+1:IF TBOM=7 THEN TBOM=0:GOSUB 3680'  INCREASE TIMER
1980     GOTO 1940
1990 '
2000 '—— END ROUTINE ——
2010 END
2020 '
2030 '▬▬ SUB ROUTINE ▬▬
2040 '—— GAME OVER ——
2050 LINE (5,5)-(33,19),"■",2,BF:LINE (6,6)-(32,18),"■",6,BF
2060 COLOR 1:FOR I=0 TO 3
2070           LOCATE 8,7+I:PRINT GM$(I);:LOCATE 8,12+I:PRINT OV$(I);
2080 NEXT
2090 COLOR 7:LOCATE 8,17:PRINT CAUSE$(CAUSE);'   PRINT CAUSE
2100 FOR I=0 TO 4000:NEXT:RETURN
2110 '
2120 '—— GAME REPLAY ——
2130 WIDTH 40,25:CONSOLE 0,25,0,1:COLOR 5,32,0:PRINT CHR$(12);
2140 GOSUB 2440'                                    MOVE TITLE
2150 COLOR 7:LOCATE 2,1:PRINT TITLE$(0)
2160         LOCATE 2,2:PRINT TITLE$(1)
2170 COLOR 1:LOCATE 2,3:PRINT TITLE$(2)
2180 COLOR 5:LOCATE 2,4:PRINT TITLE$(3):FOR J=0 TO 300:NEXT
2190 COLOR 3:LOCATE 0,0:PRINT STRING$(39,"=");'  PRINT TITLE
2200 COLOR 3:LOCATE 0,5:PRINT STRING$(39,"=");:PRINT:PRINT
2210 COLOR 6:PRINT "<<";:COLOR 7:PRINT " ﾙｰﾙ ";:COLOR 6:PRINT ">>":PRINT
2220 COLOR 7:PRINT " 1. ";:COLOR 5:PRINT "KEY        UP";:COLOR 7:PRINT 8
2230 COLOR 5:PRINT "          LEFT ";:COLOR 7:PRINT "4";
2240 COLOR 1:PRINT "——+——";:COLOR 7:PRINT "6 ";:COLOR 5:PRINT "RIGHT"
2250 COLOR 7:PRINT "                  2 ";:COLOR 5:PRINT "DOWN"
2260 COLOR 7:PRINT " 2. ";:COLOR 5:PRINT "DOT・ ";
2270 COLOR 7:PRINT "ｦ ｹｽﾄ ";:COLOR 2:PRINT "+10"
2280 COLOR 7:PRINT " 3. ";:COLOR 2:PRINT "REDCAR ";
2290 COLOR 7:PRINT "ﾄ ｼｮｳﾄﾂ ｽﾙﾄ MYCAR ﾊ CRASH"
2300 COLOR 7:PRINT " 4. ";:COLOR 3:PRINT "■";:COLOR 4:PRINT "=BOMB"
2310 COLOR 2:PRINT "     REDCAR ";:COLOR 7:PRINT "ﾄ MYCAR ｶﾞ ｵﾅｼﾞ ｺｰｽ=CRASH"
2320 COLOR 2:PRINT "     REDCAR ";:COLOR 7:PRINT "ﾄ MYCAR ｶﾞ ﾁｶﾞｳ ｺｰｽ=+10 ﾃﾝ"
2330 LOCATE 2,20:PRINT "HI-SCORE:";HISCR
2340 LOCATE 2,21:PRINT "   SCORE:";SCR
2350 LOCATE 2,22:PRINT "     1————————GAME START"
2360 LOCATE 2,23:PRINT "     2————————GAME END":COLOR 5
2370 LOCATE 2,24:PRINT "              ・・・・・・・・・・HOW ??";
2380 I$=INPUT$(1):I=VAL(I$)
2390   IF I=1 THEN GAME=1:RETURN'                   GAME START
2400   IF I=2 THEN GAME=3:RETURN'                   GAME END
2410 GOTO 2370
2420 '
2430 '—— MOVE TITLE ——
2440 COLOR 7
2450 FOR I=1 TO 18
2460   LOCATE 20-I,2:PRINT MID$(TITLE$(3),19-I,I*2-1):FOR J=0 TO 10:NEXT
2470   LOCATE 20-I,3:PRINT MID$(TITLE$(0),19-I,I*2-1):FOR J=0 TO 10:NEXT
2480 NEXT:FOR J=0 TO 200:NEXT
2490 LOCATE 2,1:PRINT TITLE$(1)
2500 LOCATE 2,4:PRINT TITLE$(2):FOR J=0 TO 600:NEXT
2510 LOCATE 2,1:PRINT TITLE$(3)
2520 LOCATE 2,2:PRINT TITLE$(0)
2530 LOCATE 2,3:PRINT TITLE$(1)
2540 LOCATE 2,4:PRINT TITLE$(2):FOR J=0 TO 300:NEXT
2550 FOR I=0 TO 3
2560   LOCATE 2,1:PRINT TITLE$(I)
2570 NEXT:FOR J=0 TO 600:NEXT
2580 RETURN
2590 '
2600 '—— CHECK HIGH SCORE ——
2610 IF SCR>HISCR THEN HISCR=SCR'                   GET HIGH-SCORE ?
2620 RETURN
```

```
2630 '
2640 '—— SET CAR START POSITION ——
2650 GAME=0:DOT=129:BOM=1:TBOM=0
2660 GOSUB 2880'                                  ｶﾞﾒﾝ
2670 RESTORE 5400'                                SET DOT
2680 FOR I=0 TO 3
2690   FOR J=0 TO 49
2700     READ DOT(I,J)
2710 NEXT J,I:GOSUB 2800'                         SET BOMB
2720 XMYCAR=29:YMYCAR=22:C1=0:C2=0 :MCRS=2'       RESET MYCAR
2730 XRED=1    :YRED=22   :C3=0:C4=35:RCRS=2'     RESET REDCAR
2740 MGO=1:RGO=INT(RND(1)*10)+5'                  GO FLAG SET
2750   COLOR 7:LOCATE XMYCAR,YMYCAR:PRINT "◆";
2760   COLOR 2:LOCATE XRED,YRED:PRINT "●";
2770 RETURN
2780 '
2790 '—— SET BOMB ——
2800 COLOR 3'                                     COLOR OF ■
2810 I=INT(RND(1)*3+1):DOT(I, 4)=2:LOCATE 29-I*2,14:PRINT "■";'  ｺｰﾅｰ 1
2820 I=INT(RND(1)*4)   :DOT(I,16)=2:LOCATE 18,I*2+1 :PRINT "■";'  ｺｰﾅｰ 2
2830 I=INT(RND(1)*4)   :DOT(I,29)=2:LOCATE I*2+1,10 :PRINT "■";'  ｺｰﾅｰ 3
2840 I=INT(RND(1)*4)   :DOT(I,41)=2:LOCATE 12,23-I*2:PRINT "■";'  ｺｰﾅｰ 4
2850 RETURN
2860 '
2870 '—— ｶﾞﾒﾝ ——
2880 COLOR 5:PRINT CHR$(12);'                     PRINT
2890 FOR Y=1 TO 23
2900   X=(Y MOD 2)+1
2910   FOR X=X TO X+28 STEP 2
2920     LOCATE X,Y:PRINT "･";
2930 NEXT X,Y
2940 FOR I=0 TO 8 STEP 2'                         PRINT RECTANGLE
2950   IF (I MOD 8)=0 THEN COLOR 1 ELSE COLOR 5
2960   LOCATE I,I:PRINT "┌";
2970   Y1=I:Y2=24-I
2980   FOR X=I+1 TO 29-I
2990     LOCATE X,Y1:PRINT "─";:LOCATE X,Y2:PRINT "─";
3000   NEXT
3010   LOCATE X,Y1:PRINT "┐";:LOCATE X,Y2:PRINT "┘";
3020   X1=I:X2=30-I
3030   FOR Y=I+1 TO 23-I
3040     LOCATE X1,Y:PRINT "│";:LOCATE X2,Y:PRINT "│";
3050   NEXT
3060   LOCATE X1,Y:PRINT "└";
3070 NEXT
3080 X1=9 :X2=21:Y1=9 :Y2=15:GOSUB 3220'          CROSS OUT
3090 X1=14:X2=16:Y1=1 :Y2=23:GOSUB 3220
3100 X1=1 :X2=29:Y1=11:Y2=13:GOSUB 3220
3110 RESTORE 4850'                                PRINT TITLE
3120 FOR Y=1 TO 24
3130   READ I$:COLOR 4:LOCATE 31,Y:PRINT "◣";:COLOR 7:PRINT I$;
3140   COLOR 4:PRINT "◢";
3150 NEXT
3160 COLOR 2:LOCATE 31,0:PRINT "●●●●●●●●"
3170 LOCATE 10,10:PRINT "LEFT:    ";:GOSUB 3290'  PRINT MESSAGE
3180 COLOR 4:LOCATE 13,12:PRINT     "SCORE";:GOSUB 3360
3190 COLOR 4:LOCATE 10,14:PRINT "HI-SCORE";:GOSUB 3420
3200 RETURN
3210 '
3220 '—— CLEAR RECTANGLE ————————
3230 ' ——— PARA IN:X1,X2,Y1,Y2 ——
3240 FOR Y=Y1 TO Y2
3250   FOR X=X1 TO X2
3260     LOCATE X,Y:PRINT " ";
3270 NEXT X,Y:RETURN
3280 '
3290 '—— PRINT LEFT ——
3300 LOCATE 15,10:PRINT "    ";'                  ERASE LAST MESSAGE
3310 IF LEFT<=1 THEN RETURN'                      LEFT=0 THEN ﾋｮｳｼﾞ ｾｽﾞ
3320 FOR X=15 TO 14+LEFT-1
3330   COLOR 7:LOCATE X,10:PRINT "◆";
3340 NEXT:RETURN
3350 '
3360 '—— PRINT SCORE ---
3370 SCR$=RIGHT$("0000"+RIGHT$(STR$(SCR),LEN(STR$(SCR))-1),5)
3380 COLOR 7:LOCATE 15,13:PRINT SCR$;
3390 DOT=DOT-1:IF DOT=0 THEN GAME=1'              DOT=0 ?
3400 RETURN
3410 '
3420 '—— PRINT HI-SCORE ——
3430 HISCR$=RIGHT$("0000"+RIGHT$(STR$(HISCR),LEN(STR$(HISCR))-1),5)
```

```
3440 COLOR 7:LOCATE 15,15:PRINT HISCR$;:RETURN
3450 '
3460 '—— CHANGE DIRECTION MYCAR ——
3470 XMYCAR=XMYCAR-1:YMYCAR=YMYCAR+1:MCRS=1:RETURN'     FOR RIGHT
3480 XMYCAR=XMYCAR+1:YMYCAR=YMYCAR+1:MCRS=2:RETURN'     FOR UP
3490 XMYCAR=XMYCAR+1:YMYCAR=YMYCAR-1:MCRS=3:RETURN'     FOR LEFT
3500 XMYCAR=XMYCAR-1:YMYCAR=YMYCAR-1:MCRS=4:RETURN'     FOR DOWN
3510 '
3520 '—— MOVE MYCAR ——
3530 COLOR 5:LOCATE XMYCAR,YMYCAR:PRINT " ";'     ERASE MYCAR
3540   CMEMO=CRS(C1,C2)'                          CMEMO=ｼﾝﾛ
3550   IF CMEMO=99 THEN C2=C2+1:GOTO 3540'        SKIP 99
3560   IF DOT(C1,C2)<>2 THEN 3580'                BOMB ?
3570     IF C1=C3 THEN 3870 ELSE DOT(C1,C2)=1
3580   IF DOT(C1,C2)=1 THEN I=USR4(0):SCR=SCR+10:GOSUB 3370'  SCORE COUNT UP
3590   DOT(C1,C2)=0'                              ERASE DOT
3600   C2=C2+1:IF C2=50 THEN C2=0'                1-ROUND END ?
3610   ON CMEMO GOSUB 3470,3480,3490,3500,4020,4080,4140,4200,4260,4320
3620   XMYCAR=XMYCAR+XADD(MCRS):YMYCAR=YMYCAR+YADD(MCRS)
3630 COLOR 7:LOCATE XMYCAR,YMYCAR:PRINT "◆";'     PRINT MYCAR
3640 IF XRED=XMYCAR AND YRED=YMYCAR THEN 3870
3650 RETURN
3660 '
3670 '—— INCREASE TIMER ——
3680 I=USR5(0)'                                   SOUND
3690 COLOR 2:LOCATE 31,25-BOM:PRINT " ▲■▲ ";
3700 BOM=BOM+1:IF BOM=25 THEN GAME=2:CAUSE=2
3710 RETURN
3720 '
3730 '—— MOVE REDCAR ——
3740 IF DOT(C3,C4)=2 THEN COLOR 3 ELSE COLOR 5'  FOR TRACE REDCAR
3750 LOCATE XRED,YRED:PRINT TRACE$(DOT(C3,C4));' ERASE REDCAR
3760   MEMO=CRS(C3,C4)'                           MEMO=ｷｭｳ ｼﾝﾛ
3770   C4=C4-1:IF C4<0 THEN C4=49'                ROUND END ?
3780   CMEMO=CRS(C3,C4)'                          CMEMO=ｼﾝﾛ
3790   IF CMEMO=99 THEN C4=C4-1:GOTO 3780'        SKIP 99
3800   ON CMEMO GOSUB 3970,3960,3990,3980,4380,4440,4500,4560,4620,4680
3810   XRED=XRED+XADD(RCRS):YRED=YRED+YADD(RCRS)
3820 COLOR 2:LOCATE XRED,YRED:PRINT "●";'         PRINT REDCAR
3830 IF XRED=XMYCAR AND YRED=YMYCAR THEN 3870
3840 RETURN
3850 '
3860 '—— CRASH MYCAR ——
3870 FOR I=1 TO 6
3880   OUT 81,33:J=USR3(0)
3890   OUT 81,32:FOR J=1 TO 20:NEXT
3900 NEXT
3910 LEFT=LEFT-1:GOSUB 3300'                      PRINT LEFT OF MYCAR
3920 IF LEFT<=0 THEN GAME=2:CAUSE=1 ELSE GAME=1' GAME OVER PRINT
3930 RETURN
3940 '
3950 '—— CHANGE DIRECTION REDCAR ——
3960 XRED=XRED+1:YRED=YRED+1:RCRS=3:RETURN'       FOR LEFT
3970 XRED=XRED-1:YRED=YRED+1:RCRS=2:RETURN'       FOR UP
3980 XRED=XRED-1:YRED=YRED-1:RCRS=1:RETURN'       FOR RIGHT
3990 XRED=XRED+1:YRED=YRED-1:RCRS=4:RETURN'       FOR DOWN
4000 '
4010 '—— GO RIGHT? MYCAR ——
4020 GOSUB 4730:IF KY$<>"R" THEN RETURN'          KEY OFF
4030   XMYCAR=XMYCAR+2'                           CHANGE LOCATE
4040   IF MCRS=2 THEN C1=C1-1 ELSE C1=C1+1'              CORSE
4050 RETURN
4060 '
4070 '—— GO UP? MYCAR ——
4080 GOSUB 4730:IF KY$<>"U" THEN RETURN'          KEY OFF
4090   YMYCAR=YMYCAR-2'                           CHANGE LOCATE
4100   IF MCRS=1 THEN C1=C1+1 ELSE C1=C1-1'              CORSE
4110 RETURN
4120 '
4130 '—— GO LEFT? MYCAR ——
4140 GOSUB 4730:IF KY$<>"L" THEN RETURN'          KEY OFF
4150   XMYCAR=XMYCAR-2'                           CHANGE LOCATE
4160   IF MCRS=2 THEN C1=C1+1 ELSE C1=C1-1'              CORSE
4170 RETURN
4180 '
4190 '—— GO DOWN? MYCAR ——
4200 GOSUB 4730:IF KY$<>"D" THEN RETURN'          KEY OFF
4210   YMYCAR=YMYCAR+2'                           CHANGE LOCATE
4220   IF MCRS=1 THEN C1=C1-1 ELSE C1=C1+1'              CORSE
4230 RETURN
4240 '
```

```
4250 '—— GO LEFT OR RIGHT? MYCAR ——
4260 GOSUB 4730'                                        KEYSCAN
4270   IF KY$="L" THEN 4150'                            LEFT
4280   IF KY$="R" THEN 4030'                            RIGHT
4290 RETURN
4300 '
4310 '—— GO UP OR DOWN? MYCAR ——
4320 GOSUB 4730'                                        KEYSCAN
4330   IF KY$="D" THEN 4210'                            LEFT
4340   IF KY$="U" THEN 4090'                            RIGHT
4350 RETURN
4360 '
4370 '—— GO RIGHT? REDCAR ——
4380 IF C3=C1 OR MEMO<>0 THEN RETURN
4390   XRED=XRED+2'                                     CHANGE LOCATE
4400   IF RCRS=2 THEN C3=C3+1 ELSE C3=C3-1'                    CORSE
4410 RETURN
4420 '
4430 '—— GO UP? REDCAR ——
4440 IF C3=C1 OR MEMO<>0 THEN RETURN
4450   YRED=YRED-2'                                     CHANGE LOCATE
4460   IF RCRS=1 THEN C3=C3-1 ELSE C3=C3+1'                    CORSE
4470 RETURN
4480 '
4490 '—— GO LEFT? REDCAR ——
4500 IF C3=C1 OR MEMO<>0 THEN RETURN
4510   XRED=XRED-2'                                     CHANGE LOCATE
4520   IF RCRS=2 THEN C3=C3-1 ELSE C3=C3+1'                    CORSE
4530 RETURN
4540 '
4550 '—— GO DOWN? REDCAR ——
4560 IF C3=C1 OR MEMO<>0 THEN RETURN
4570   YRED=YRED+2'                                     CHANGE LOCATE
4580   IF RCRS=1 THEN C3=C3+1 ELSE C3=C3-1'                    CORSE
4590 RETURN
4600 '
4610 '—— GO LEFT OR RIGHT? REDCAR ——
4620 IF C3=C1 OR MEMO<>0 THEN RETURN
4630   IF C3>C1 THEN C3=C3-1:IF RCRS=2 THEN XRED=XRED-2  ELSE XRED=XRED+2
4640   IF C3<C1 THEN C3=C3+1:IF RCRS=2 THEN XRED=XRED+2  ELSE XRED=XRED-2
4650 RETURN
4660 '
4670 '—— GO UP OR DOWN? REDCAR ——
4680 IF C3=C1 OR MEMO<>0 THEN RETURN
4690   IF C3>C1 THEN C3=C3-1:IF RCRS=1 THEN YRED=YRED-2  ELSE YRED=YRED+2
4700   IF C3<C1 THEN C3=C3+1:IF RCRS=1 THEN YRED=YRED+2  ELSE YRED=YRED-2
4710 RETURN
4720 '
4730 '—— KEYSCAN ——
4740 I0=INP(0):I1=INP(1):KY$=""'                        GET CHECK KEY ON
4750 IF I0<>255 OR I1<>255 THEN 4770'                   KEY ON
4760   IF RND(1)*100<95 THEN RETURN ELSE MGO=INT(RND(1)*10)+1:RETURN'   STOP
4770   IF I0=239 THEN KY$="L":RETURN'                   LEFT ON
4780   IF I0=191 THEN KY$="R":RETURN'                   RIGHT ON
4790   IF I0=251 THEN KY$="D":RETURN'                   DOWN ON
4800   IF I1=254 THEN KY$="U":RETURN'                   UP ON
4810 RETURN
4820 '
4830 '▬▬▬ DATA AREA ▬▬▬
4840 '—— TITLE DATA ——
4850 DATA "■▔▔▔◣"
4860 DATA "■▁▁▁◤"
4870 DATA "■   ◣"
4880 DATA "■▁▁▁◤"
4890 DATA "◢▔▔▔◣"
4900 DATA "■   ■"
4910 DATA "■   ■"
4920 DATA "◥▁▁▁◤"
4930 DATA "◣   ◢"
4940 DATA "■◣▁◢■"
4950 DATA "■ ▔ ■"
4960 DATA "■   ■"
4970 DATA "■▔▔▔◣"
4980 DATA "■▁▁▁◤"
4990 DATA "■   ◣"
5000 DATA "■▁▁▁◤"
5010 DATA "■▔▔▔◥"
5020 DATA "■▁▁ ."
5030 DATA "■    "
5040 DATA "■▁▁▁◢"
```

```
5050 DATA "    "
5060 DATA "    "
5070 DATA "    "
5080 DATA "    "
5090 '
5100 DATA "                    ";"                "
5110 DATA "                    ";"                "
5120 DATA "                    ";"                "
5130 DATA "                    ";"                "
5140 '
5150 '—— COURSE DATA ——
5160 DATA 00,00,00,00,07,07,00,00,00,00,03:'                  COURSE-0
5170 DATA 00,00,00,00,00,08,08,08,00,00,00,00,00,04
5180 DATA 00,00,00,00,05,05,00,00,00,00,01
5190 DATA 00,00,00,00,00,06,06,06,00,00,00,00,00,02
5200 DATA 00,00,00,99,09,09,00,00,00,99,03:'                  COURSE-1
5210 DATA 00,00,00,00,99,10,10,10,00,00,00,00,99,04
5220 DATA 00,00,00,99,09,09,00,00,00,99,01
5230 DATA 00,00,00,00,99,10,10,10,00,00,00,00,99,02
5240 DATA 00,00,99,99,09,09,00,00,99,99,03:'                  COURSE-2
5250 DATA 00,00,00,99,99,10,10,10,00,00,00,99,99,04
5260 DATA 00,00,99,99,09,09,00,00,99,99,01
5270 DATA 00,00,00,99,99,10,10,10,00,00,00,99,99,02
5280 DATA 00,99,99,99,05,05,00,99,99,99,03:'                  COURSE-3
5290 DATA 00,00,99,99,99,06,06,06,00,00,99,99,99,04
5300 DATA 00,99,99,99,07,07,00,99,99,99,01
5310 DATA 00,00,99,99,99,08,08,08,00,00,99,99,99,02
5320 '
5330 '—— ADD DATA ——
5340 DATA +2, 0:'                                   RIGHT
5350 DATA  0,-2:'                                   UP
5360 DATA -2, 0:'                                   LEFT
5370 DATA  0,+2:'                                   DOWN
5380 '
5390 '—— INITIAL DOT DATA ——
5400 DATA 1,1,1,1,1,0,1,1,1,1,1:'                   COURSE-0
5410 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
5420 DATA 1,1,1,1,1,0,1,1,1,1,1
5430 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
5440 DATA 1,1,1,1,1,0,1,1,1,1,1:'                   COURSE-1
5450 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
5460 DATA 1,1,1,1,1,0,1,1,1,1,1
5470 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
5480 DATA 1,1,1,1,1,0,1,1,1,1,1:'                   COURSE-2
5490 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
5500 DATA 1,1,1,1,1,0,1,1,1,1,1
5510 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
5520 DATA 1,1,1,1,1,0,1,1,1,1,1:'                   COURSE-3
5530 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
5540 DATA 1,1,1,1,1,0,1,1,1,1,1
5550 DATA 1,1,1,1,1,1,0,0,1,1,1,1,1,1
5560 '
5570 '—— MACHINE LANGUAGE DATA (CC00-CDA0) ——
5580 DATA AF,D3,51,11,FF,35,21,2A,CD,CD,88,CC,16,0A,01,40
5590 DATA 20,CD,5F,CC,15,20,F7,18,0C,AF,D3,51,11,FF,25,21
5600 DATA 83,CD,CD,88,CC,01,19,28,C3,3A,09,01,50,30,C3,5F
5610 DATA CC,01,10,10,CD,51,CC,C9,00,00,00,00,01,05,10,C3
5620 DATA 51,CC,21,90,CD,7E,A7,C8,46,23,4E,23,CD,51,CC,18
5630 DATA F4,F5,3E,FF,CD,6F,CC,3D,20,FA,0D,20,F5,F1,C9,CD
5640 DATA 6F,CC,04,0D,20,F9,C9,CD,6F,CC,05,0D,20,F9,C9,D5
5650 DATA 50,3A,67,EA,CB,EF,D3,40,15,20,FB,50,3A,67,EA,CB
5660 DATA AF,D3,40,15,20,FB,D1,C9,7E,A7,C8,47,23,4E,07,30
5670 DATA 05,CD,A9,CC,18,03,CD,DE,CC,3A,67,EA,CB,AF,D3,40
5680 DATA 06,10,CD,22,CD,10,FB,18,DF,E5,D5,CD,B5,CC,D1,0D
5690 DATA 20,F8,E1,23,C9,EB,50,3A,67,EA,CB,AF,D3,40,2B,7C
5700 DATA A7,28,15,15,20,F8,50,3A,67,EA,CB,AF,D3,40,2B,7C
5710 DATA A7,28,05,15,20,F8,18,DE,3A,67,EA,CB,AF,C9,E5,D5
5720 DATA CD,EA,CC,D1,0D,20,F8,E1,23,C9,EB,50,3A,67,EA,CB
5730 DATA EF,D3,40,2B,7C,A7,28,15,15,20,F8,50,3A,67,EA,CB
5740 DATA AF,D3,40,2B,7C,A7,28,05,15,20,F8,18,DE,C9,CD,22
5750 DATA CD,10,FB,C9,D5,11,FF,FF,1D,C2,18,CD,15,C2,18,CD
5760 DATA D1,C9,D5,1E,FF,1D,20,FD,D1,C9,22,02,22,02,1E,02
5770 DATA 1E,02,22,03,26,01,28,04,22,02,22,02,1E,02,1E,02
5780 DATA 22,03,26,01,28,04,22,02,22,02,1E,02,1B,02,19,04
5790 DATA 16,04,1B,04,1B,01,1E,01,1B,01,1E,01,22,04,1E,02
5800 DATA 1B,02,19,04,19,02,1E,02,22,08,1E,04,22,02,26,02
5810 DATA 28,08,22,02,22,02,33,02,26,02,28,04,2D,04,33,0E
5820 DATA FF,02,00,33,01,2D,01,28,01,22,02,28,01,22,04,00
5830 DATA 32,32,4B,32,32,32,4B,32,64,32,96,32,4B,32,14,32
5840 DATA 00
```

《GAMING最前線》

アドベンチャーゲームへの招待

㈲マイクロキャビン ＆ ㈲アローソフト
大矢知直登 代表

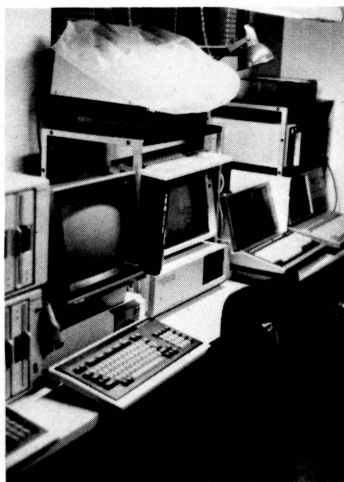
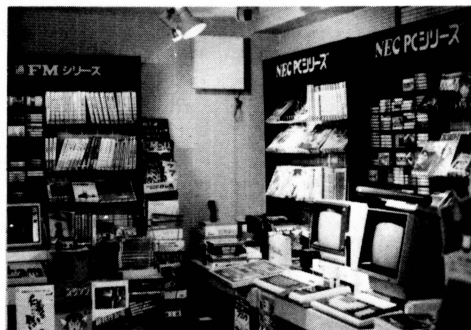
▲ミステリーハウス

▲大矢知代表

昨年の5月にＤＩＳＫ版のアドベンチャーゲーム，ミステリーハウスを発表して以来，ミステリーハウスII，ダイヤモンドアドベンチャー，ＴＨＥ　ＳＰＹ，最新作ドリームランドなど入門クラスのものから，上級向きのアドベンチャーゲームを開発してきましたが，最近では，いろんなストーリーのアドベンチャーゲームが発表されて，日本においてもパソコンゲームの一つのジャンルを確立した感があります。

ミステリーハウスの発売当時は，アドベンチャーゲームという名称自体が，なじみがうすくて，ユーザーの方からの〝ゲームの進め方が全然理解できない！〟とか，〝答えを教えろ！〟等，質問があいつぎ，我々の方も電話の応対で業務に支障をきたしたりして，それならばということで，ミステリーハウスIIには詳しいヒントをつけて販売すると，今度は，アドベンチャーゲームにヒントを付けるとはけしからんなどとおしかりを受けたりして，私供の開発スタッフも振回された時期もありました（質問のほうは現在でも多いのですが…）。

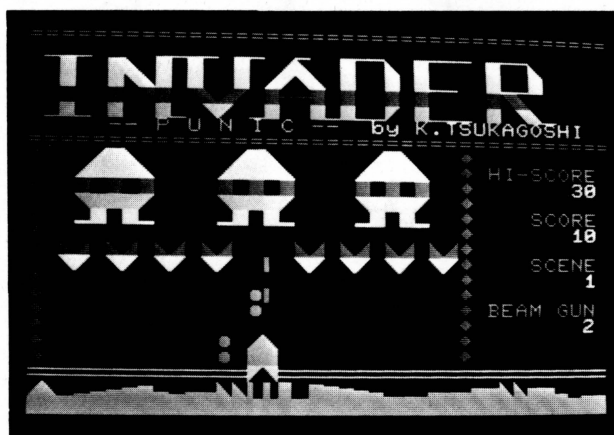
アドベンチャーゲームは，本来，ある目的（例えば宝石を見つける等）を達成する為にＣＲＴ上に現れる情景やコメントからヒントを得て，自分で推理する事を楽しむゲームではないかと思います。その意味においても，ヒントや答えを製作者に聞くことはルール違反ではないでしょうか。素晴らしいアドベンチャーゲームを開発中の他のソフトハウスの方々の為にも，そういう行為は自重していただきたいと思います。アドベンチャーゲームは，じっくりと楽しむものですから。

◀ソフト開発室

さて，このアドベンチャーゲームを実際に作る場合，一番苦労するのが，〝いかに難しく，しかも解きやすく〟という，一見相反する要素をうまくかみ合せて作るということです。この点を克服して，あとはグラフィックの美しさが加味されますと，一級品のアドベンチャーゲームの出来上りというわけです。しかし，とにかく普通のリアルタイムゲームより製作に時間がかかります。そして最後にプログラムのデバッグが待ちうけています。これがなかなかの曲者です。ひどい時には，プログラムの製作以上に時間と手間がかかる場合もあります。まあ，こんな風にして，我々のスタッフ（名前を上げますと，Dr. Moritani，H. Hashimoto，N. Minami，K. Ito，etc）が悪戦苦闘してアドベンチャーゲームを製作しています。スタッフ一同，今後とも新作のアドベンチャーゲームに挑戦していきますので御期待下さい！

▲マイクロキャビン店内

'インベーダー・パニック'に挑戦

20世紀フォックス「スター・ウォーズ／ジェダイの復讐」より

「インベーダーパニック」に挑戦

プロローグ

はじめに

名曲喫茶を様変わりさせたり，国鉄職員を盗みに走らせたり，百円玉が不足して日銀をあわてさせるとの話しまで出たり――インベーダーゲームは，盛り場はもちろんおふろ屋さんやガード下にまではびこり，まさに日本列島を侵略しつくそうとしているが，最近，東京の中高生らがインチキ硬貨や電子ライターを使ってまんまと機械を動かし，その不正がばれるという例が相次いでいる。精巧さを売り物にした機械の意外な弱みにつけこむ"〝インベーダー〟たちに，業者は神経をぴりぴり。警視方は，もう放置できないと監視強化を宣言した。「子どもは遊びの天才」といわれるが，未成年者をそこまで駆りたてる過熱ブームに，疑問を寄せる声が上がっている。

(朝日新聞1979年５月17日)

〝スペース・インベーダー〟が登場したのが，1978年の秋のこと。そして1979年にかけて日本中にパニックを起こしたことは，良く御存知の通りです。そして御記憶にあるように，

はやる遊びは世にたたかれる

の習いにしたがい，衰退の道も早かったわけです。

しかし，この **〝インベーダー・パニック〟**が，マイコンGAMEに残した影響は凄じく，そのスタイルまでもまったく変革してしまいました。**〝スペース・インベーダー〟の影響を受けていないGAMEはない，**と言って良いくらいです。したがって，GAME作りをマスターする上で，

スペース・インベーダー

は格好の教材となるでしょう。

〝GAMINGへの招待〟でも，第４ブロックとしてこの

スペース・インベーダー

否，**ビーム攻撃型リアルタイムGAME**を取り上げてみることにします。題して

インベーダー・パニック

はじまりはじまり。

インベーダー・パニック？

――この間，マイコン教室に行って来ましたよ。

――マイコン教室？　ホウ，今流行の。

――そう，マイコン買って，もうずいぶんと時間が経った。人のプログラムばかり入れているんじゃ申し分けない，と思ってね。

――一大奮起ですな。それで，いくらか出来るようになりましたか？

――いや，それで先生様に聞いてみました。

――ホウ，ホウ。

――先生，私は自分のマイコンを買って随分になります。一生懸命マイコンにむかってガチャガチャやっているんですが，一向にプログラムらしいものが出来ません。一体，どういうわけなんでしょう？

――それで先生，何と答えました？

――先生様がおっしゃるには，

※スペース・インベーダーは(株)タイトーが開発，同社が権利を有するゲームです。

「良くそうおっしゃる方がいます。確かにそういう方はマイコンの前に座り，良く努力しておられる。しかし，唯マイコンの前に座るだけではダメなんですねえ。プログラムを作るには，やはりきちんと構想をたてなければいけません。きちんと構想が出来たら，**フローチャート**に書いてみる。マイコンにむかうのは，それからでも遅くありませんねえ」

——なる程ねえ。それでうまく行きましたか？

——いや，ちゃんとフローチャートを書こうとしたんだけど，うまくいきません。

——それで？

——また先生様におうかがいをたてました。

——ホウ，ホウ。

——先生様がおっしゃるには，

「良くそうおっしゃる方がいるんですよ。プログラムの作れない人に話しを聞くと，大体初めからフローチャートを作ろうとする。頭で考えているんですね。それよりも，まず機械にむかうことですよ。御自分の機械をお持ちでしょう？どんどん，キーボードをたたいてごらんなさい。すぐプログラムなんて出来上がりますよ」

何だかどうしていいんだかわからなくなりました。

——なる程，さすがに先生だ。いいことをおっしゃる。何事も中間，**中間**がいいんですなあ。

——ハハア，中間ね。わかった！　それで私の機械のＢＡＳＩＣ，**中間言語**を使っているんだな。

——ホワイト，キック（白蹴る——シラケル）。

〝インベーダー・パニック〟って何でしょう？　どうせ**ビーム攻撃型ＧＡＭＥ**を取り上げるのなら，本家本元の〝スペース・インベーダー〟を取り上げてもらいたいものですね。

たしかに〝スペース・インベーダー〟は良く出来たＧＡＭＥです。**〝スペース・インベーダー〟をマスターすれば，ほとんどのリアルタイムＧＡＭＥを作れるようになるでしょう。**

ところでその〝スペース・インベーダー〟,あまりに良く出来ているため，あちこちに面白い仕掛けがしてあります。そこでもしこれを〝ＧＡＭＩＮＧへの招待〟で取り上げようとすると，非常に長くなってしまうのです。すでに長いプログラムを経験した人なら良いのですが，まだＧＡＭＥ作りを始めて間もない人ですと，**途中で息切れ**してしまう恐れがあります。

そこで〝スペース・インベーダー〟に登場するテクニックはすべて取り入れ，かつもう少し手軽に作れるＧＡＭＥを考えてみました。それが，

インベーダー・パニック

です。とは言え，〝ＧＡＭＩＮＧへの招待〟でのプログラムは，その性格上，いつも原稿と同時進行で作っていくため，**現段階では，どんなＧＡＭＥになるのかわかっていません**。コワイですね。〝スペース・インベーダー〟よりも難しくなったりして。マ，どんなＧＡＭＥになるか，御期待ください。また，〝スペース・インベーダー〟についても，いつかは〝ＧＡＭＩＮＧへの招待〟で取り入れたいと思っています。ハイ，ハイ，ハイ。

インデックスの製作

さあ，本題に入りましたよ。ＧＡＭＥ作りが，始まりますよ！

まず，どこから作り始めましょうか？

〈注〉プログラム作りにおいて，本当は**どこから設計していったら良いか，**すでに研究されているのです。それについては，また別の機会に発表したいと思います。なにせ型にはまらないところが，〝ＧＡＭＩＮＧへの招待〟の特徴なのですから。

何はともあれ，まず**プログラムのタイトル**を作っておきましょう。

リスト４—１を御覧ください。

ＲＥＭ文による**タイトルの覚え書き**です。雑誌等でおなじみですね。ここに日付等を入れておくと，それがあなたの記念になります。

ところで私の初期のプログラムには，ＲＥＭが入っていません。と言うのは，当時のマシンはＲＡＭ容量が少なく，とても**ＲＥＭ文なんか入れる余裕がなかったの**です。なにせスタートレックを作ろうにも，途中でパンクして作れなかった位ですから。

したがって当時のプログラムを見ても，いつ作ったんだか，本当に自分が作ったんだかわからなくなっています。残念なことをしました。

そこであなたも，**リスト４—１**のようなＲＥＭ文を作ってみてください。カッコだけでも，**上級のプログラマー**になったような気がしますよ。あとは，このＲ

《リスト４—１》ＲＥＭ文によるタイトルの覚え書き

```
1000 '================================
1010 '   INVADER PANIC --list 1--
1020 '        1982.5.4-?.??
1030 '                by K.TSUKAGOSHI
1040 '================================
```

ＥＭ文に続いてどんなプログラムが続くか次第です。ハイ，ハイ。

〈糾弾コーナー〉

？？：ＲＥＭと言っていますが，**リスト４－１**にはＲＥＭ文が出ていませんよ。それに行番号のあとの’は，何ですか？

ツカ：ゴミです――イ，イヤ。**アポストロフィ**です。ＰＣ－8001の場合，注釈文は

REM

’

どちらも使えます。使い方や使用ＲＡＭ容量の違いはありますが，**REM**よりは’の方が目立たないので’を使ってみました。

？？：それだけ？

ツカ：まだ何か言うのですか。エート，……。

ちなみにアセンブラで注釈文をつけるときは，；（**セミコロン**）を使う場合が多いです。ＦＯＲＴＲＡＮでは，１字目に**C**があるとその行は注釈文とみなされます。

？？：それだけ？

ツカ：大型機のＯＳでは，いちいち**リスト４－１**のような部分を作らなくても，自動的に作成されます。それも花文字かなんかで。

？？：それだけ？

ツカ：降参！（もう，ちっとも先に進まなくなっちゃう！）

画面モードの設定

次は，自分のマシンのモードの設定を行ってください（**リスト４－２**）。私の場合，

画面サイズ＝40×25
カラー・モード　｝1110行

《リスト４－２》画面モードの設定

```
1000 '==============================
1010 '   INVADER PANIC --list 2--
1020 '        1982.5.4-?.??
1030 '               by K.TSUKAGOSHI
1040 '==============================
1050 '
1060 '---------
1070 '   MAIN
1080 '---------
1090 '
1100 '== COLD START ==
1110 WIDTH 40,25:CONSOLE 0,25,0,1    'SET TV MODE
1120 PRINT CHR$(12);                 'CLEAR
```

を選択しました。もしあなたのマシンが，特にそういう設定が不要でしたら，何もする必要はありません。1120行は，画面クリアの命令です。この部分は，どなたも自分のマシンに合わせて，入れておいてください。

これでプログラムが２行だけできました。**リスト４－２**をＲＵＮすると，画面がクリアされておしまいになります。

〈糾弾コーナー〉

？？：何も有りません。

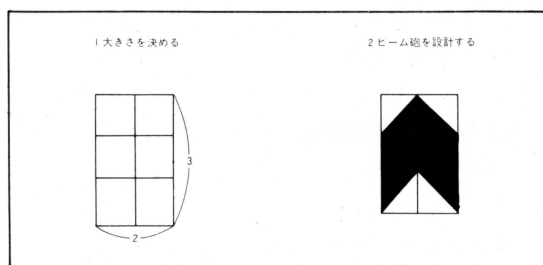
ツカ：そうでしょう。

ビーム砲の設計

続いて**第３段階**――ボチボチ，**ビーム砲の設計**あたりに入っていきましょう。

ビーム砲：インベーダー攻撃のための唯一の武器

です。慎重に設計しましょう。

まず画面サイズ（私のでしたら40×25）から考えて，どの位の大きさにするか決めます（**第４－１図①**）。

《第４－１図》ビーム砲の設計

２×３

の大きさで描くことにしました。大きさが決まれば，このワク内に収まるようにビーム砲を作っていきます。手持ちのキャラクタを，いかに組合わせるかが勝負です。**キャラクタの設計**が，ＧＡＭＥ作りの中でも一番楽しく，また重要な部分であるかもしれません。気に入ったものが出来ないときは，**大きさ**を変えたり，**グラフィック**に変更したり，あきらめたりします。

いちおう**第４－１図②**の形で行くことにしました。何，ビーム砲に見えない？　スミマセンネエ。

設計が終わりましたら，プログラ

ム化です。

PRINT　"▲"
PRINT　"■"
PRINT　"◥◤"

で表示されますね。これを取り入れたのが，**リスト4—3**です。また，これを走らせたのが，**写真1**です。

〈糾弾コーナー〉

？？：**リスト4—3**の1140～1160行が，ビーム砲の表示ですね？

ツカ：そうです。

？？：各行，ビーム砲の**左右に空白**があいていますが，何か意味があるのですか？

ツカ：気まぐれです。イヤ，イヤ，ちゃんと**魂胆**があるのです。この空白は，あとで必要になります。その時，説明致しましょう。

？？：エラそうに。

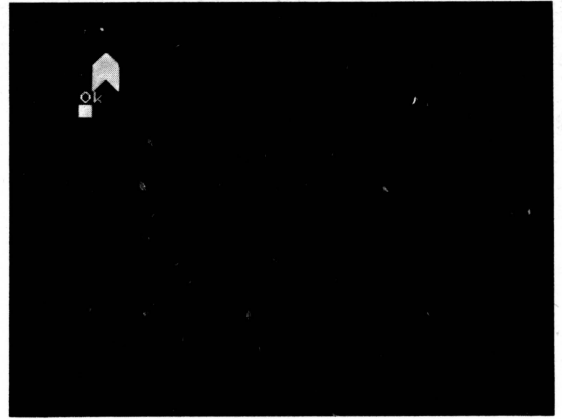
PC—8001では，カラーが使えますので，せっかくですからビーム砲にもカラーをつけておきましょう。

COLOR　色番号

色番号	
1	青
2	赤
3	紫
4	緑
5	水色
6	黄
7	白

このカラー命令によって色をつけたのが，**リスト4—4**です。これ走らせると，**写真1**が〝色つきのビーム砲″になります。

(注)　配色については，他とのバランスから将来予告なしに変更することがあります（無節操！）。

《写真1》ビーム砲の表示

《リスト4—3》ビーム砲の表示

```
1000 '================================
1010 '   INVADER PANIC --list 3--
1020 '       1982.5.4-?.??
1030 '              by K.TSUKAGOSHI
1040 '================================
1050 '
1060 '---------
1070 '   MAIN
1080 '---------
1090 '
1100 '== COLD START ==
1110 WIDTH 40,25:CONSOLE 0,25,0,1   'SET TV MODE
1120 PRINT CHR$(12);                'CLEAR
1130 '
1140 PRINT " ▲ "
1150 PRINT " ■ "
1160 PRINT " ◥◤ ";
```

《リスト4—4》ビーム砲に色をつける

```
1000 '================================
1010 '   INVADER PANIC --list 4--
1020 '       1982.5.4-?.??
1030 '              by K.TSUKAGOSHI
1040 '================================
1050 '
1060 '---------
1070 '   MAIN
1080 '---------
1090 '
1100 '== COLD START ==
1110 WIDTH 40,25:CONSOLE 0,25,0,1   'SET TV MODE
1120 PRINT CHR$(12);                'CLEAR
1130 '
1140 COLOR 1:PRINT " ▲ "
1150 COLOR 5:PRINT " ■ "
1160 COLOR 1:PRINT " ◥◤ ";
```

自由な，自由な位置に

そろそろ第1章の終りに近づきました。次の章では，

ビーム砲を動かす

等，**キースキャン**の問題を取り上げることになります。それには，このセクションにおいて，〝ビーム砲の表示ルーチン″をもう少ししっかりした形にしておく必要があります。

リスト4—3なり，**リスト4—4**をRUNして〝ビーム砲″はどこに表示されましたか？　**いつも画面の左上**に表示されましたね。これでは，

自由にビーム砲を動かす

ことはできません。ビーム砲を

任意の位置

に表示できなくてはダメなのです。

これから**リスト４－４**のビーム砲表示部分（1140～1160行）を，上の要求に答える形（サブルーチン）に書き換えてみましょう。

パラメータ

それは，プログラムの構造上，次のようになるでしょう。

```
（メインルーチン）
        ∫
   〔表示位置の指定〕
    GOSUB  （ビーム砲表示）
        ∫
```

```
（ビーム砲表示ルーチン）
    指定された位置にビーム砲を表示する
```

すなわちメインルーチンは，

〝どこに表示したいか〟という情報

をサブルーチンに渡します。サブルーチンは，その情報を見て初めて仕事にかかります。この**ルーチン間で渡される情報**を

パラメータ

と呼んでいます。

サブルーチンにパラメータを渡す方法は，いろいろありますが，ＢＡＳＩＣの場合は特にそのような機能はありません。そこで通常

メモリを仲介

にパラメータを渡すことになります。

メモリと言ってもびっくりすることはありません。ＢＡＳＩＣでは**物理的なメモリの位置**を気にする必要はなく，**論理的なメモリ（変数）**を自由に使えます。論理上のメモリ位置と物理的なメモリ位置との変換は，**インタプリタ**が自動的に行ってくれます。

プログラム化すると

それでは具体的な話しに戻ります。

メインルーチンとサブルーチンでパラメータの受け渡しをするため，

ビーム砲の位置を表わす変数

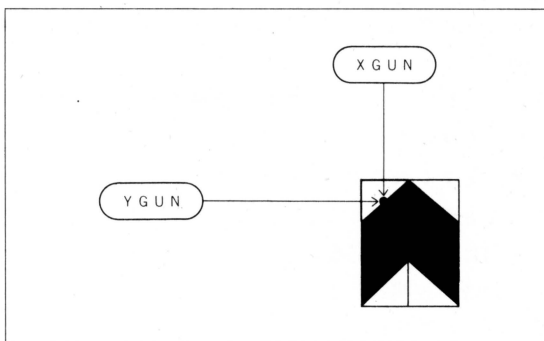
を用意します。ここでは

ＸＧＵＮ：ヨコ座標

ＹＧＵＮ：タテ座標

を採用することにします。ここで，〝**ビーム砲の位置**〟とは，

ビーム砲の左肩（第４－２図）

を指すものとします。

《第４－２図》ビーム砲の位置

これでパラメータが決まりました。たとえば，

（２０，１０）

の位置にビーム砲を表示したいと思ったら，そのイメンルーチンは次のようになるでしょう。

```
XGUN＝２０：YGUN＝１０
GUSUB ××
      ↑
  （〝ビーム砲表示ルーチン〟の行番号）
```

また，それに応える〝ビーム砲表示ルーチン〟は，

```
LOCATE XGUN, YGUN：
  PRINT 〝 ▲ 〟
LOCATE XGUN, YGUN＋１：
  PRINT 〝 ■ 〟
LOCATE XGUN, YGUN＋２：
  PRINT 〝 ◣◢ 〟
RETURN
```

となるでしょう。このパラメータの受け渡しを図示すれば，**第４－３図**のようになります。

モジュールの独立宣言

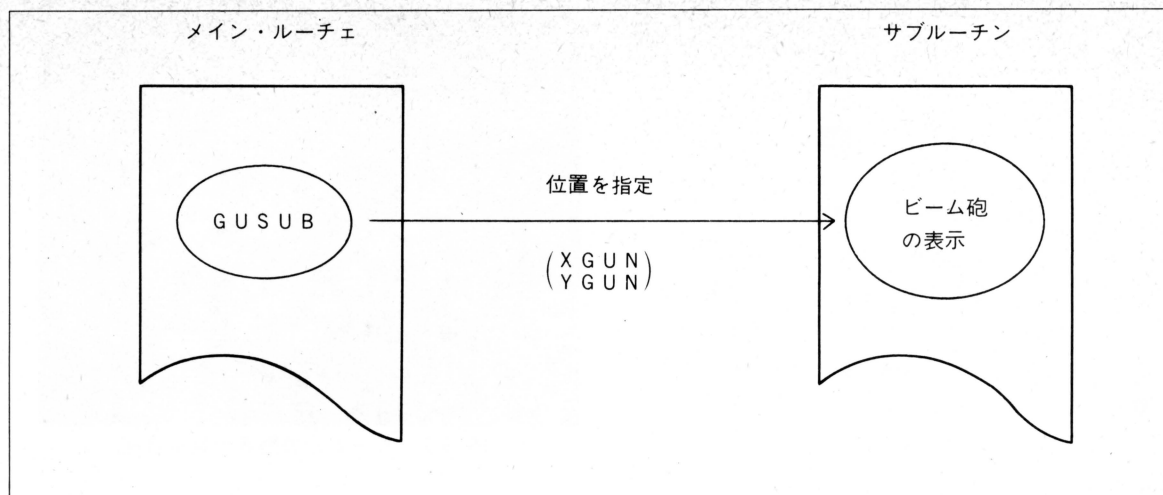
以上の構想のもとに

〝ビーム砲の表示ルーチン〟を

サブルーチン化

したのが，**リスト４－５**です。

メインルーチンを見てみましょう。

1150行が，**パラメータの設定**をしているところです。

《第４－３図》パラメータを渡す

《リスト４－５》ビーム砲の表示のサブルーチン化

```
1000 '=============================
1010 '   INVADER PANIC --list 5--
1020 '        1982.5.4-?.??
1030 '               by K.TSUKAGOSHI
1040 '=============================
1050 '.
1060 '--------
1070 '   MAIN
1080 '--------
1090 '
1100 'XGUN,YGUN  :LOCATE OF BEAM GUN
1110 '
1120 '== COLD START ==
1130 WIDTH 40,25:CONSOLE 0,25,0,1              'SET TV MODE
1140 PRINT CHR$(12);                           'CLEAR
1150 XGUN=18:YGUN=22
1160 GOSUB 1240                                'CALL PRINT BEAM GUN
1170 GOTO 1170                                 'INSTEAD OF END
1180 '
1190 '-------
1200 '   SUB
1210 '-------
1220 '
1230 '== PRINT BEAM GUN ==
1240 COLOR 1:LOCATE XGUN,YGUN   :PRINT " ▲ "
1250 COLOR 5:LOCATE XGUN,YGUN+1:PRINT " ■ "
1260 COLOR 1:LOCATE XGUN,YGUN+2:PRINT " ▞ ";
1270 RETURN
```

ここでは，

（１８，２２）

の位置に表示しようとしています。そして1160行で〝ビーム砲表示ルーチン〞をＣＡＬＬしています。

〝ビーム砲表示ルーチン〞は，1240～1270行です。御覧のように前のリストを少し変えて，サブルーチン化してあります。

〈糾弾コーナー〉

ツカ：何もないでしょう？

？？：あります！

ツカ：ど，どうぞ。

？？：ビーム砲は，横にしか動きませんね？

ツカ：ハイ。

？？：それなら**ビーム砲のＹ座標は，常に固定**されているわけですよね？　つまりＹ座標は不変なわけですから，パラメータは**ＸＧＵＮだけで良く，**ＹＧＵＮまで指定する必要はなかったのではないのですか？　さらに言えば，たとえば

ＹＧＵＮ＝２３

とすれば，1240～1260行は（ＣＯＬＯＲ文を抜かすと），

LOCATE XGUN, 22:
　　PRINT "▲"
LOCATE XGUN, 23:
　　PRINT "■"
LOCATE XGUN, 24:
　　PRINT "◣◢"

と，ずっと簡単になったのではないですか？

ツカ：御説，ごもっともです。そして，もちろんそれでかまいません。ただし，サブルーチンをそのように作ってしまうと，ビーム砲の動きが**その行を左右に動くだけ**に限定されてしまい，

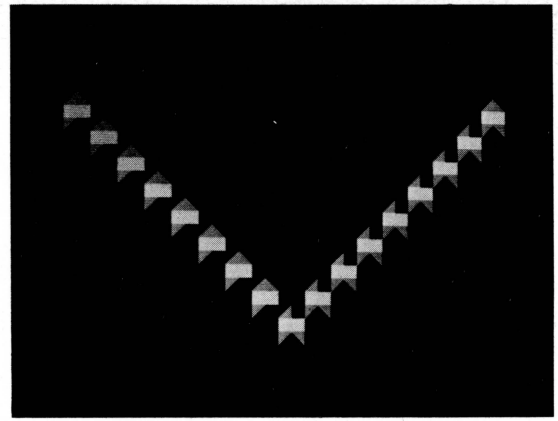
自　由　度

がなくなってしまいます。たとえば，

① ビーム砲の位置をもう少し上に変更したい。
② デモンストレーションでビーム砲を上下に自由に動したい。

ということが起こっても，**？？さん**のように組んでしまうと，沢山の行を変更したり，新たにサブルーチンを作ったりしなければなりません。

？？：フーン。

ツカ：私のようにサブルーチンを作っておけば，**写真2**のようなことも容易にできます。これ，なかなか面白いでしょう？　参考にこの画面を得るためのプログラムを**リスト4―6**に示しておきます。

《写真2》ビーム砲の動きが自由自在

？？：フーン。

ツカ：プログラム作成時の鉄則として，

各モジュール（プログラフの単位）は
出来るだけ独立させる！

というのがあります。プログラムの各モジュールを

《リスト4―6》ビーム砲の動きを自由自在にする

```
1000 '==============================
1010 '   INVADER PANIC --list 6--
1020 '        1982.5.4-?.??
1030 '               by K.TSUKAGOSHI
1040 '==============================
1050 '
1060 '--------
1070 '   MAIN
1080 '--------
1090 '
1100 'XGUN,YGUN  :LOCATE OF BEAM GUN
1110 '
1120 '== COLD START ==
1130 WIDTH 40,25:CONSOLE 0,25,0,1                'SET TV MODE
1140 PRINT CHR$(12);                             'CLEAR
1141 '
1150 FOR XGUN=0 TO 15 STEP 2                     'LINE UP BEAM GUN FORM-V
1160   YGUN=XGUN:GOSUB 1300
1170 NEXT
1180 XGUN=XGUN+1:GOSUB 1300
1190 FOR XGUN=16 TO 32 STEP 2
1200   YGUN=32-XGUN:GOSUB 1300
1210 NEXT
1211 '
1220 GOTO 1220                                   'INSTEAD OF END
1230 '
1250 '-------
1260 '   SUB
1270 '-------
1280 '
1290 '== PRINT BEAM GUN ==
1300 COLOR 1:LOCATE XGUN,YGUN  :PRINT " ▲ "
1310 COLOR 5:LOCATE XGUN,YGUN+1:PRINT " ■ "
1320 COLOR 1:LOCATE XGUN,YGUN+2:PRINT " ◣◢ ";
1330 RETURN
```

このように作っておけば，あとでプログラムを修正する必要が起こっても，

変更点を最少にとどめることができる

わけです。

第1章のおわりに

本セクションでは，二つのパラメータを導入することで自由な動きを見せる

ビーム砲表示ルーチン

の製作まで進みました。ためしに**リスト4―5**の1150行で

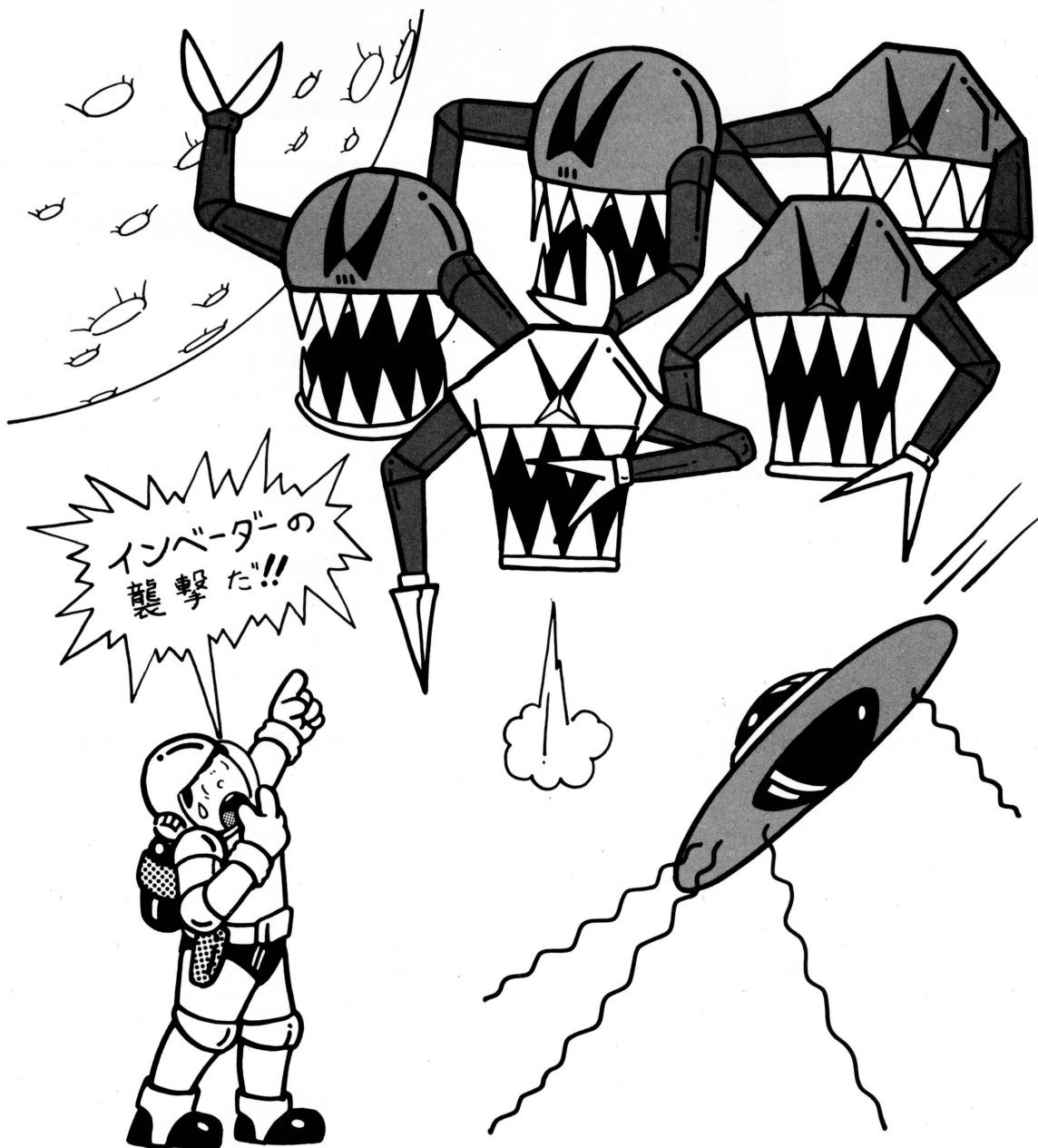
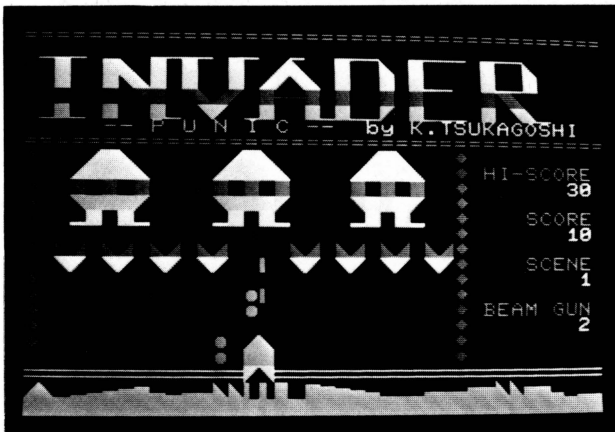
XGUN，YGUN

の値をいろいろ変えてRUNしてみてください。

次の第2章では，キースキャンの問題を取り上げ，いよいよ

ビーム砲を動かす

ことに挑戦します。難しそうだ？　いや，だいじょうぶです。第2章以降，楽に進めるか否かは，私の原稿の書き方次第にかかっていますから。そして，あなたの**多大な努力**と――。

「インベーダー・パニック」に挑戦

リアルタイムキー入力に挑戦

は　じ　め　に

日が長くなるから，夏が近づく
夏が近づくと，暑くなる
暑くなると，汗が出る
汗が出るから，センスであおぐ
あおぐと，汗が出る
汗が出るから，友達おだててあおがせる
　(だんだん涼しくなる)
〝ありがとう。おかげで涼しくなりました。
私の汗は，ひっこみました。
オヤ？　どうしたのですか？　汗が出ていますね。この涼しいのに〃

《力学的汗保存の法則》より

だんだん夏が近づき，暑い日が続くようになってきました。そろそろ第2章，始めることにしましょうか？

キーの配置

第1章では，自由な位置に**ビーム砲**を表示できる。

ビーム砲表示ルーチン

を作りあげました（**リスト4—5**)。今度は，その続きです（**リスト4—5**は，お遊びですから無視)。

これから取り上げるのは，

キースキャン

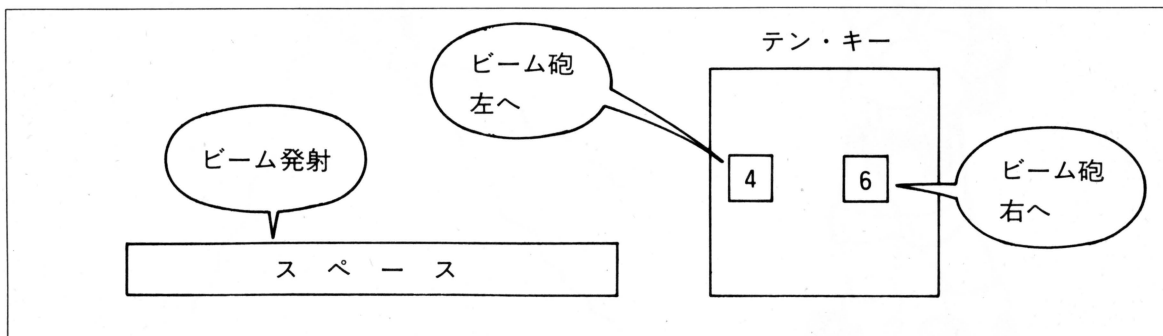
の問題です。すなわち

ある特定のキー

が押されているかをキャッチし，もしそうならば**それに応じた処理**をさせようとするものです。

キーの種類は，**二つ**あります。

① **キー4**

このキーが押されているなら，**ビーム砲を左に移動**します。

《第4—4図》キー・ボードの配置

② **キー6**

このキーが押されているなら，**ビーム砲を左に移動**します。

③ **スペース・キー**

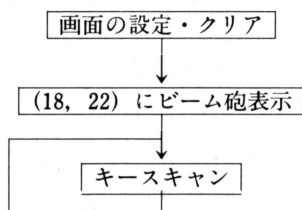
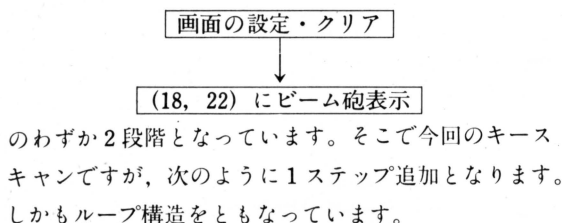
このキーが押されているなら，**ビームを発射**します。

以上のようにこの〝インベーダー・パニック〟では，ゲーム進行に**三つのキー**を使います。そこでこれらのキーを**キー・ボード**に配置することになります。私のPC—8001では，**第4—4図**のように配置しました。あなたもあなたのマシン上に，これらのキーを配置してみてください。**人間工学的に使いやすい位置に。**

メインルーチンの変更

次に**メインルーチンの流れ**を見てみましょう。

復習のため最初に**リスト4—5**の場合を見てみると，

のわずか2段階となっています。そこで今回のキースキャンですが，次のように1ステップ追加となります。しかもループ構造をともなっています。

この流れを理解していただくため，実際のプログラムを御覧いただきましょう。**リスト4—7**です。

ここで御注意申し上げておきますが，プログラムは行番号1000から**リナンバー**をかけてあります。そこで上の流れと対応させると，次のようになります。

1140〜1150：画面の設定・クリア
1160〜1170：ビーム砲の表示
1190：キースキャン
1200：ループを作る

すなわち1190行でCALLしている1330行からのサブルーチンが，キースキャン処理を行っている部分といえます。さあ，そうすると，

サブルーチン：KEY SCAN

とはどんな処理を行っているのでしょう？　それがこれからの焦点となります。

リアルタイム・キー入力

ところで一般に〝キースキャン〟の問題は，そのマシンのハードウエアに依存するため，どうしてもマシンによって扱いが異なってしまいます。そこで各自のマシンの

リアルタイム・キー入力

の方法を調べていただくことになります。すなわち，先にあなたが設定したキーが

押されているか・いないか

を判定する命令を調べていただきたいのです。これはあなたのマシンのマニュアルに書かれているはずです。

さて，ここではPC—8001におけるキー入力キャッチの仕方を簡単に説明しておくことにします。さもないと，**サブルーチンKEY SCAN**を理解していただくことができず，先に進めないからです。

PC—8001でキーの入力をキャッチするには，

INP（引数）

という関数を使います。引数は，

0〜255

までの値が使えます。このうちキーボードのキャッチのために使うのは，

0〜9

までで，キャッチしたい**キーの種類**によっていくつを使うかは決まっています。今回必要なのは，次の三つです。

キー4（ビーム砲左）——INP（0）
キー6（ビーム砲右）——INP（0）
スペース（ビーム発射）——INP（9）

したがって，

INP（0）
INP（9）

の二つの値を調べれば良いということになります。

次にその**INPの値**についてです。

まずキーが押されていないときは，

INP（引数）＝255

です。1340行を御覧ください。**二つのINP関数がともにゼロ**ということは，**4**も**6**も**スペース**も**押されていない**ということです。このときは，何もしないでRETURNしています。

キーが押されると，INPの値は255から変化します。それはキーの種類により異なります。

キー4：INP（0）＝239
キー6：INP（0）＝191

《リスト４－７》KEK SCANルーチンの追加

```
1000 '============================
1010 '  INVADER PANIC --list 7--
1020 '       1982.5.4-?.??
1030 '                  by K.TSUKAGOSHI
1040 '============================
1050 '
1060 '--------
1070 '  MAIN
1080 '--------
1090 '
1100 'XGUN,YGUN  :LOCATE OF BEAM GUN
1110 'I0,I9       :VALUE OF INP
1120 '
1130 '== COLD START ==
1140 WIDTH 40,25:CONSOLE 0,25,0,1              'SET TV MODE
1150 PRINT CHR$(12);                           'CLEAR
1160 XGUN=18:YGUN=22
1170 GOSUB 1270                                'CALL PRINT BEAM GUN
1180 COLOR 7                                   'CHANGE COLOR TO WHITE
1190 GOSUB 1330                                'KEY SCAN
1200 GOTO 1190
1210 '
1220 '-------
1230 '  SUB
1240 '-------
1250 '
1260 '== PRINT BEAM GUN ==
1270 COLOR 1:LOCATE XGUN,YGUN  :PRINT " ▲ "
1280 COLOR 5:LOCATE XGUN,YGUN+1:PRINT " ■■ "
1290 COLOR 1:LOCATE XGUN,YGUN+2:PRINT " ▛▜ ";
1300 RETURN
1310 '
1320 '== KEY SCAN ==
1330 I0=INP(0):I9=INP(9)                       'KEY SCAN
1340 IF I0=255 AND I9=255 THEN RETURN          'NO TOUCH THEN RET
1350 IF I0=239 THEN 1410                       'LEFT ?
1360 IF I0=191 THEN 1450                       'RIGHT ?
1370 IF I9=191 THEN 1490                       'SPACE ?
1380 RETURN                                    'OTHE KEY THEN RET
1390 '
1400 '== MOVE LEFT BEAM GUN ==
1410 LOCATE 10,10:PRINT "LEFT "
1420 RETURN
1430 '
1440 '== MOVE RIGHT BEAM GUN ==
1450 LOCATE 10,10:PRINT "RIGHT"
1460 RETURN
1470 '
1480 '== SHOOT BEAM GUN ==
1490 LOCATE 10,10:PRINT "SPACE"
1500 RETURN
```

スペース・キー：ＩＮＰ（９）＝191

以上が，ＰＣ－8001によるリアルタイムキー入力の予備知識です。これで1320行～1380行のサブルーチンが読めると思います。ここの処理をまとめると，**第４－５図**のようになるのがわかるでしょう。

処理の流れ

リアルタイム・キー入力がわかったところで，もう一度**プログラムの流れ**を振りかえってみましょう。

メインルーチンでは，画面の初期設定，画面のクリアを行ったのち，初期位置にビーム砲を表示します。そして問題のＫＥＹ ＳＣＡＮをＣＡＬＬします。**ＫＥＹ ＳＣＡＮ**では，キーが押されていなければすぐＲＥＴＵＲＮしてきます。キーが押されていれば，キーの種類に応じてそれぞれのサブルーチンに処理の依託をします。たとえばもし[4]のキーが押されていたなら，ビーム砲を一つ左に移すという具合に。**メインルーチン**では，ＫＥＹ ＳＣＡＮから戻ってくると，またＫＥＹ ＳＣＡＮをＣＡＬＬします。こうして永久に同じサブルーチンを呼び続け，**リスト４－７**のプログラムは，

《第4—5図》KEY SCANの処理

LOCATE　10, 10

でカーソルを所定の位置にもっていってから,

PRINT　"×××××"

としていますね。

ここでは, 2点程御注意しておきましょう。

① 1180行の

COLOR　7

(表示を白にする)

は, 1170行でビーム砲を表示しはあと,

LEFT, RIGHT,

SPACE

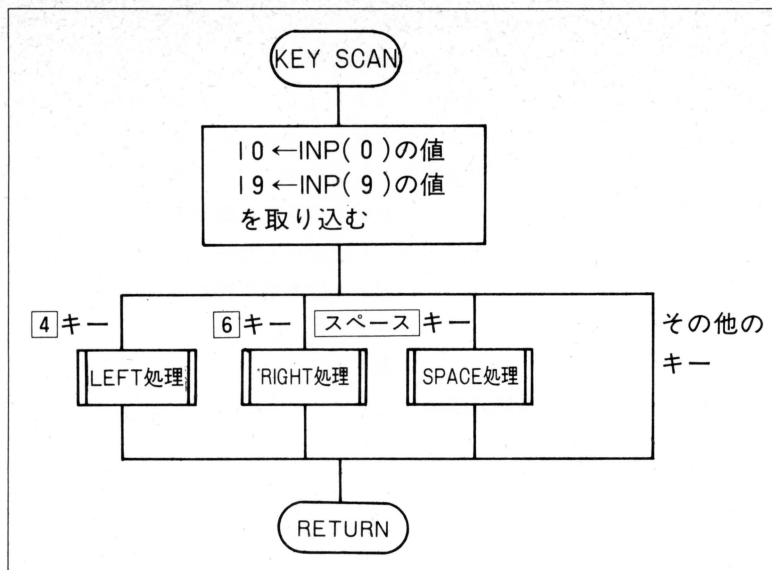
を白で表示するためです。

② 1410行で**LEFTのあとのスペースは**, 他の文字 (RIGHT, SPACE) を表示したあとで, **5文字目を消去**するためです。

以上で**リスト4—7**は, すべておわかりになったことと思います。

さっそくプログラムを走らせてみましょう。**写真3**は, SPACE キーを押したところです。

SPACE

の文字が見えますね。

終ることがありません。

さあ, これで**プログラムの流れ, KEY SCANのしくみ**が大体おわかりいただけたことと思います。あとは,

1400行：ビーム砲を左へ

1440行：ビーム砲を右へ

1480行：ビーム発射

のルーチンを作れば良いわけです。しかし, ここでそれをやっていると長くなってしまい, 焦点がボケてしまいます。そこでKEY SCANルーチンがうまく動くかを確認するため,

4が押された

⟶ "LEFT　" を表示

6が押された

⟶ "RIGHT" を表示

SPACEが押された

⟶ "SPACE" を表示

するように, とりあえず仮のプログラムを作ってみることにしました。

リスト4—7の完成

それでは, **リスト4—7**を御覧になってください。

1400～1420："LEFT　" 表示

1440～1460："RIGHT" 表示

1480～1500："SPACE" 表示

になっているのがわかると思います。それぞれのサブルーチンは,

《写真3》SPACE キーを押すと

次の目標は？

リスト4—7をRUNさせてみて, いかがでしたか？　ガチャ, ガチャ,

4, 6, SPACE
のキーを押してみましたか？　そのたびに

LEFT

RIGHT

SPACE

の文字が，次々に変わっていくのがおわかりになると思います。これはつまり

サブルーチン：KEY SCAN

がうまく働いた証拠であるわけです。したがって我々は安心して次に進むことができます。

さあ，今度はいよいよ

ビーム砲を動かす！

ことに挑戦してみましょう。すなわち，次の二つのサブルーチンを作り変えれば良いのです。

〝LEFT　〞を表示したルーチン
——→**ビーム砲を左に動かす**ルーチン

〝RIGHT〞を表示したルーチン
——→**ビール砲を右に動かす**ルーチン

この二つのルーチンは，

左と右

《リスト4—8》ビーム砲の移動

```
1000 '============================
1010 '   INVADER PANIC --list 8--
1020 '        1982.5.4-?.??
1030 '                by K.TSUKAGOSHI
1040 '============================
1050 '
1060 '--------
1070 '   MAIN
1080 '--------
1090 '
1100 'XGUN,YGUN   :LOCATE OF BEAM GUN
1110 'I0,I9       :VALUE OF INP
1120 '
1130 '== COLD START ==
1140 WIDTH 40,25:CONSOLE 0,25,0,1            'SET TV MODE
1150 PRINT CHR$(12);                          'CLEAR
1160 XGUN=18:YGUN=21
1170 GOSUB 1260                               'CALL PRINT BEAM GUN
1180 GOSUB 1320                               'KEY SCAN
1190 GOTO 1180
1200 '
1210 '-------
1220 '   SUB
1230 '-------
1240 '
1250 '== PRINT BEAM GUN ==
1260 COLOR 1:LOCATE XGUN,YGUN   :PRINT " ▲ "
1270 COLOR 5:LOCATE XGUN,YGUN+1:PRINT " ■ "
1280 COLOR 1:LOCATE XGUN,YGUN+2:PRINT " ◤◥ ";
1290 RETURN
1300 '
1310 '== KEY SCAN ==
1320 I0=INP(0):I9=INP(9)                      'KEY SCAN
1330 IF I0=255 AND I9=255 THEN RETURN         'NO TOUCH THEN RET
1340 IF I0=239 THEN 1400                      'LEFT ?
1350 IF I0=191 THEN 1440                      'RIGHT ?
1360 IF I9=191 THEN 1480                      'SPACE ?
1370 RETURN                                   'OTHE KEY THEN RET
1380 '
1390 '== MOVE LEFT BEAM GUN ==
1400 IF XGUN<1 THEN RETURN                    'LEFT EDGE ?
1410 XGUN=XGUN-1:GOTO 1250
1420 '
1430 '== MOVE RIGHT BEAM GUN ==
1440 IF XGUN>34 THEN RETURN                   'RIGHT EDGE ?
1450 XGUN=XGUN+1:GOTO 1250
1460 '
1470 '== SHOOT BEAM GUN ==
1480 LOCATE 10,10:PRINT "SPACE"
1490 RETURN
```

の違いだけですから，最初にビーム砲を左に動かすルーチンを考えてみることにします。右に動かすルーチンは，それをほんの少し変えるだけでできますから。ハイ。

ビーム砲を左に動かす

どうしたらビーム砲が，左に動くか考えてみましょう。

リスト4—8を御覧ください。1160～1170行を見ればわかるように，ビーム砲は最初

(18, 21)

の位置に表示されます。これを模型的に描いたのが，**第4—6図**①です。図では，斜線の部分がビーム砲を表わしています。ここでもし，

XGUN＝XGUN－1

としたらどうなるでしょうか？　XGUNは，**ビーム砲のヨコ座標**を表わしていますから，

XGUN＝17

となり，ビーム砲の位置が

一つ左に移動する

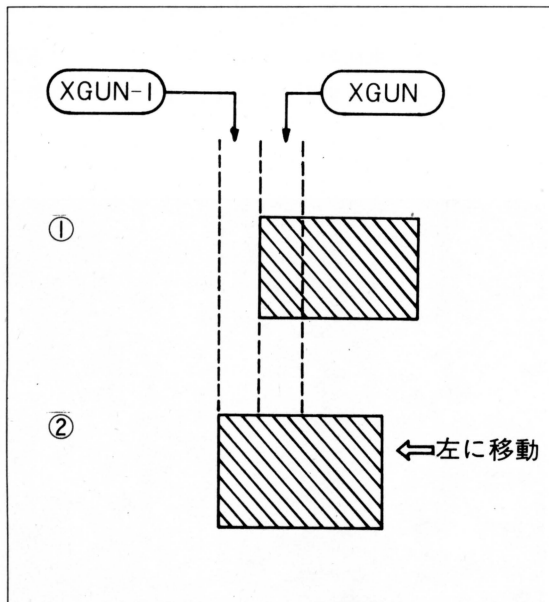
ことになります（**第4—6図**②)。これがすなわち

ビーム砲を動かす原理

です。

さあ，これで

ビーム砲を左に動かすルーチン

では，次の二つの処理を行えばよいことがおわかりになったと思います。

《第4—6図》ビーム砲移動の原理

XGUNの値を一つ小さくする

↓

その位置にビーム砲を書く

リスト4—8の1410行を見てください。上の一つの処理が行われているのがわかりますね？

〈糾弾コーナー〉

？？：大体，ビーム砲を左に動かす原理はわかりました。でもこのやり方だと

ビーム砲の右端が残ってしまう！

のではないですか（**第4—7図**)？

ツカ：ハア。鋭いところをついていますね。前の**〈糾弾コーナー〉**で，**ビーム砲のキャラクタの左右に空白が入れてある**問題が出ましたね？

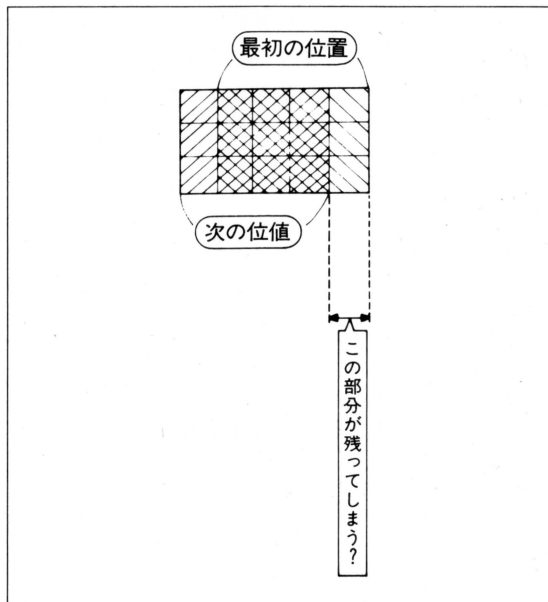
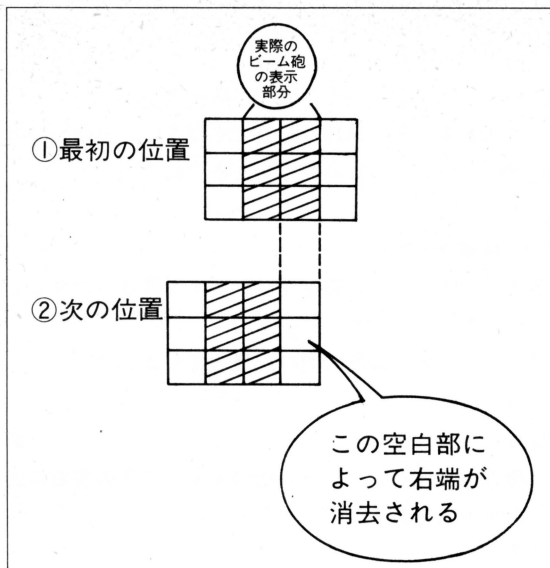
？？：——。

ツカ：今，その部分が活躍するときがきました。ビーム砲の移動については，**第4—6図**のように説明しました。しかし実際は，**第4—8図**のようになっているのです。つまり①のように，**ビーム砲の左右が空白**になっています。次にこのビーム砲を左に動かすと，②のようになります。そしてこの右側の空白のようになります。そしてこの右側の空白のおかげで，前のビーム砲はすべて消されてしまいます。

？？：ホウ。

ツカ：左側の空白は，ビーム砲が右に動くとき役に立つわけです。OK？

？？：——。感心，感心。

ツカ：（ニヤリ)。

《第4—7図》ビーム砲の右端が残る

《第4—8図》右端消去の原理

はじっこは、恐いね

さあ，もう問題はないでしょうか？

サブルーチンKEY SCANがCALLされ，そのとき4のキーが押されていると，1390行の

ビーム砲を左に移動するルーチン

に飛び込みます。そしてビーム砲を一つ左に動かすと，メインルーチンに戻ってきます。そしてまたサブルーチンKEY SCANがCALLされて——。

ところでこの処理が繰り返されると，一体どういうことが起こるでしょうか？

ビーム砲が一つずつ左に移動していく。一つずつ。そして，やがて——。そうです！ このまま放っておくと，やがて**ビーム砲は画面をはみだし，**

エラー！

になってしまうのです。

そこでこの

画面はみ出しチェック

を行う必要があります。どうしたら良いでしょうか？

〔A君〕：左端に来たら，4を押すのをやめる。
——ゲームになりません。

〔B君〕：左端に来たら，電源コードを抜く。
——かなりの反射神経を要求されます。

〔C君〕：〝エラー〞が出たら，
RUN＋↗
と押す。
——神経を疑います。

〝GAMINGへの招待〞では，やはりきちんと**ソフトで対応**しましょう。やり方は，簡単ですよ。

ビーム砲のX座標は，

XGUN

でわかります。もし，

XGUN＝0

であれば，もう**ビーム砲は左端にきている**ことになります。ですから，このときは何もしないですぐに

RETURN

してやることにします。**リスト4—8**の1400行を御覧ください。これがビーム砲の左端をチェックしているところです。

ビーム砲が動いた

以上でビーム砲を動かすルーチンは，おしまいです。よくリストを御覧になって納得してください。また，〝ビーム砲を右に動かすルーチン〞もまったく同様に作れますから，合わせて研究してみてください。

それでは，これからできあがった**リスト4—8**を走らせてみることにします。

RUN＋↗

写真4は，ビーム砲を左へ動かしたところです。画面の左の方に寄っていますね。また**写真5**は，ビーム砲を右に動したところです。スペース・キーも，もちろんききます。〝SPACE〞の文字が見えますね。やあ，できました，できました。プログラムがだいぶできあがってきましたね。

ここまできたら，次に進む前にもう少し手を入れておきましょう。画面が少し淋しいですね。ここで**画面をグッと派手に**しますよ。すると，ますますやる気が出てきます。ハイ。

《写真4》ビーム砲を左へ移動

《写真５》ビーム砲を右へ移動

《写真６》画面を豪華に

《写真７》SPACEキーを押すと

画面を豪華に

まず**写真６**を御覧ください。また**写真７**は，スペース・キーを押したものです。これらの写真は，**リスト４―９**のプログラムを走らせたものです。もしあなたのマシンがＰＣ―8001，ＰＣ―8800でしたら，**リスト４―９**を入力し，あなたのマシンで走らせてみてください。またあなたのマシンがそうでないなら，友達のＰＣに入力させ，ぜひ**実物のカラー**で御覧になってください。いかがですか？　だいぶＧＡＭＥの画面らしくなってきたでしょう？　もちろんビーム砲は，左右に動きます。スペース・キーもききます。ただしまだビームは出ませんが，そして何と感激的なことに，**あなたはこのプログラムのすべてを理解することが可能**なのです。

以下に**リスト４―９**の必要な部分を説明しておきましょう。

① **リスト４―９**は，**リスト４―８**を発展させて〝タイトル〟等をつけ加えたものです。しかもその処理のほとんどは，

ＰＲＩＮＴ文だけ

で行っています。

② サブルーチンは，**リスト４―８**とほとんど同じです。違いは，次の２点です。

- キャラクタを1470～1490行のように変更した。
- 配色を変えた。

③ **1270～1400行**

タイトルを表示している部分です。使った命令は，

ＰＲＩＮＴ

ＣＯＬＯＲ（色を指定する）

の二つだけです。

④ **1410～1440行**

画面下の都市をＰＲＩＮＴするものです。ここで注釈しておきますと，1440行の

ＬＩＮＥ文

は，マシンによっては使えないかもしれません。これは，画面の最下行を■でうめるもので，

ＰＲＩＮＴ　"■■■■■■■■■■■■"
（長さを１行分とる）

と同じです。ただし，このようにすると画面が**スクロールしてしまいます**から，右端を１字分あけるとよいでしょう。

以上のように**簡単なＰＲＩＮＴ文の追加で、画面が俄然生き生きとしてきます**。あなたもあなたのアイデアで素敵な画面を作ってみてください。

〈糾弾コーナー〉

？？：1440行をＬＩＮＥ文を使わずにすます方法がありますか？

ツカ：エヘヘ。やはりこの命令，使わない方が良かったかもしれませんね。まずもっとも簡単なのは，やはり今述べたようにＰＲＩＮＴ文を使う方法でしょ

```
1000 '============================
1010 '   INVADER PANIC --list 9--
1020 '        1982.5.4-?.??
1030 '               by K.TSUKAGOSHI
1040 '============================
1050 '
1060 '--------
1070 '   MAIN
1080 '--------
1090 '
1100 'XGUN,YGUN   :LOCATE OF BEAM GUN
1110 'IO,I9       :VALUE OF INP
1120 '
1130 '== COLD START ==
1140 WIDTH 40,25:CONSOLE 0,25,0,1                 'SET TV MODE
1150 PRINT CHR$(12);                              'CLEAR
1160 XGUN=18:YGUN=20
1170 GOSUB 1270                                   'CALL PRINT GAME ARER
1180 GOSUB 1470                                   'CALL PRINT BEAM GUN
1190 GOSUB 1530                                   'KEYSCAN
1200 GOTO 1190
1210 '
1220 '--------
1230 '   SUB
1240 '--------
1250 '
1260 '== PRINT GAME AREA ==
1270 COLOR 1
1280 PRINT "========================================"
1290 COLOR 7
1300 PRINT "  __  ___ __ _ __ ___ __              "
1310 PRINT "  ■ ■◣■■ ■ ▲■ ■◣■ ■ ■◣              "
1320 PRINT "  ■ ■◣■■ ■◢▲■ ■ ■_ ■_■              "
1330 COLOR 2
1340 PRINT "  ■ ■◥ ◥◢ ■━■■ ■■ ■◣              "
1350 COLOR 3
1360 PRINT " _■_ ■ ■_ ▼ _■ ■■_◤ ■__ ■◥_           "
1370 COLOR 6:PRINT "      -- P A N I C --   ";
1380 COLOR 7:PRINT "by K.TSUKAGOSHI "
1390 COLOR 1
1400 PRINT "========================================"
1410 LOCATE 0,22
1420 COLOR 5:PRINT "=======================================";
1430 COLOR 6:PRINT "▲___▄▄___◢◣▌■■ ■▄▄_______▄▄◢◣▄▄_ __";
1440 LINE (0,24)-(38,24),"■",6,BF
1450 '
1460 '== PRINT BEAM GUN ==
1470 COLOR 1:LOCATE XGUN,YGUN   :PRINT " ▲ "
1480         LOCATE XGUN,YGUN+1:PRINT " ■ "
1490 COLOR 5:LOCATE XGUN,YGUN+2:PRINT "◢◤◥◣";
1500 RETURN
1510 '
1520 '== KEY SCAN ==
1530 IO=INP(0):I9=INP(9)                          'KEY SCAN
1540 IF IO=255 AND I9=255 THEN RETURN             'NO TOUCH TEHN RET
1550 IF IO=239 THEN 1610                          'LEFT ?
1560 IF IO=191 THEN 1650                          'RIGHT ?
1570 IF I9=191 THEN 1690                          'SPACE ?
1580 RETURN                                       'OTHE KEY THEN RET
1590 '
1600 '== MOVE LEFT BEAM GUN ==
1610 IF XGUN<1 THEN RETURN                        'LEFT EDGE ?
1620 XGUN=XGUN-1:GOTO 1460
1630 '
1640 '== MOVE RIGHT BEAM GUN ==
1650 IF XGUN>34 THEN RETURN                       'RIGHT EDGE ?
1660 XGUN=XGUN+1:GOTO 1460
1670 '
1680 '== SHOOT BEAM GUN ==
1690 LOCATE 10,10:PRINT "SPACE"
1700 RETURN
```

う。他には，**POKE文**を使って直接VRAMに書込む方法が考えられます。これなら，1行をフルに使ってもスクロールはしません。ただし中級以上の人向けと言えましょう。

第2章のおわりに

第2章では，

キー入力のキャッチ

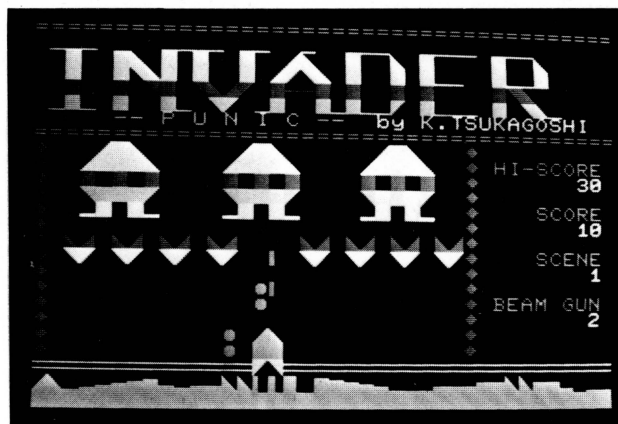
について考え，**ビーム砲の移動**に成功しました。また最後には，タイトルの文字も入り

迫力のある画面

にすることもできました。キー入力の仕方は各自のマシンにより異なりますが，その**考え方**はおわかりいただけたのではないでしょうか？

さあ，せっかくGAMEらしい画面になってきました。これからどのようなGAMEが展開されていくのでしょうか？　次章をお楽しみに。

「インベーダー・パニック」に挑戦

第3章 インベーダーが登場して

はじめに

BASIC言語にもある程度慣れ，そろそろゲームでも作ってみようということで，動きのある

　　リアルタイムGAME

に挑戦する人が増えてきました。彼等の大部分が最初にぶつかる壁は，おそらく

　　“どうやって絵を動かすの？”

ということでしょう。しかしこの壁は，わりと簡単に突破していくようです。なぜなら気の効いた参考書なら

　　簡単な絵の動かし方くらいは
　　説明されている

からで，それらを読めばわりと簡単に図形（普通は1キャラクタでしょう）を動かすことができるようになります。そして実際に自分のマシンでやってみると確かに動きます。

　　“やった！　絵が動いた！”

ということで喜びます。

　　“これで何でも作れるぞ！”

と思ったとたん，**目の前が真暗**になるのを感じるでしょう。それは大きく二つに分かれるようです。

①　**キー入力の方法**がわからない。

②　**複数の絵を同時に動かす方法**がわからない。

なぜなら，これら二つはほとんど解説されていないからです。それどころか，そのような問題があることさえ指摘されていません。

たとえば，こんな具合です。

絵の動かし方がわかったので，さっそくある図形を動かしてみます。たとえば，ビーム砲を左右に動かしてみたり。すると①に気づいた人は，

　　ビーム砲は動いているが，それが
　　キー入力と連動していない

ことを知るでしょう。ただビーム砲がかってに画面上を左右に動くだけです。これでは，ただの**デモンストレーション**にすぎません。

また②に気づいた人は，たぶんビームを発射してみようとした人でしょう。つまり，

　　ビーム砲の動かし方はわかる
　　ビームの動かし方もわかる

それらを個々に動かすことはできる。しかし，

　　二つを同時に動かすことはできない！

ことに気づくのです。

なぜ多くの参考書が，

　　絵の動かし方

を解説しておきながら，①，②については言及していないのでしょうか？　それは，もしかしたら参考書をお書きになった先生方が，**対話型リアルタイム処理**にあまり重点を置いていなかったからかもしれません。

これは，おそらく現役の多くのプロ達が**大型コンピュータで育った**からではないかと思われます。そこで行われる処理のほとんどは，

パンチ・カードによるデータの入力
↓
データの加工

↓

結果のプリンタへの出力

という流れになっています。たとえば大型機で良く使われるＦＯＲＴＲＡＮでは，出力文として

ＷＲＩＴＥ文

を用いますが，その中に**装置指定子**というのを書きます。そしてそこで使われる番号は，大抵

6

であり，これは

ラインプリンタ

を指定していることが多いようです。

現在のようなダイナミックな動きのあるゲームが発達したのは，ここ二，三年です。マイコンで安価なビデオＲＡＭが使えるようになってからで，そのルーツは，ＴＶゲームの

テニス・ゲーム

でしょうし，飛躍的にポピュラー化させたのが，あの

スペース・インベーダー

だったのです。したがって**今日の素晴しいリアルタイムＧＡＭＥ**をここまで発展させてきたのは，多くの**アマチュア・ホビースト達**であり,「マイコン誌」をはじめとするマイコン・ホビー雑誌が

マイコン文化発展

に寄与してきたわけです。

現在マイコン・グラフィックで育った若い人達が，どんどんプロの職場に進出しつつあります。やがて若い優秀な彼等が，**プログラミングやシステムのスタイル**を変えていくことでしょう。その時，ＢＡＳＩＣやプログラミングの参考書も

ダイナミックな動きのあるもの

へと変革していくことでしょう――それまでは，この拙い〝ＧＡＭＩＮＧへの招待〟の各シリーズで我慢しておきましょう。

ということで，第3章のはじまり，はじまり――。

目標：ビームの移動

前章で我々は，

ビーム砲の移動

に成功しました。まずは，満足，満足。しかし，スペース・キーを押しても

ＳＰＡＣＥ

と表示されるだけでは，面白くありませんね。やはりそこはきちんと

ビームが発射

されてほしいものです。これで我々の次の目標が決まりました。

ビーム発射，バンザーイ！

さて，一口に〝ビーム発射〟といっても，そこには**二つの異なる手続き**があることに注意してください。

①　ビームを発射する

②　ビームを上へ動かす

の二つです。両者はまったく異なる働きをします。まずリストでそれを確認してみましょう。**リスト4―10**が，ビーム移動に成功したプログラムです。

①：1730～1770行

②：1790～1850行

のように，リスト上もこれらは**別のサブルーチン**となっています。このことは，ビームの動きを頭の中でよく考えてみるとわかるとわかるでしょう。

最初は，画面上にビームが存在していません。このときは，①が働きます。そして

スペース キー

が押されたのをキャッチすると，今度は②が働きます。そして**ビームを上へ移動させていく**わけです。

以上**二つのサブルーチン**が必要なことを認識した上で，このことをプログラムで表現していくことに致しましょう。

ビーム発射ルーチンの製作

最初は，

スペース・キーのキャッチ

です。これは，先の

〝ＳＰＡＣＥ〟プリント・ルーチン

を改良することで実現できます。

まずビームの位置を表わす変数として

（ＸＢＥＡＭ，ＹＢＥＡＭ）

を導入しましょう。そしてこのうち**ビームのタテの位置**を表わす**ＹＢＥＡＭ**に注目しましょう。ＹＢＥＡＭの動ける範囲は，

19≦ＹＢＥＡＭ≦8

です（**第4―9図**）。するとこのＹＢＥＡＭを

現在ビームがあるかないか？

の判定材料に使うことができます。それには，次のようにします。

まず

ＹＢＥＭ＝0――――――――①

にセットしておきます（1190行）。すなわち①のように**ＹＢＥＡＭの値が0**のときは，

現在ビームがない

```
1000 '=============================
1010 '  INVADER PANIC --list 10--
1020 '       1982.5.4-?.??
1030 '             by K.TSUKAGOSHI
1040 '=============================
1050 '
1060 '--------
1070 '  MAIN
1080 '--------
1090 '
1100 'I0,I9         :VALUE OF INP
1110 'BMAX          :MAX OF YBEAM
1120 'XBEAM,YBEAM   :LOCATE OF BEAM
1130 'XGUN ,YGUN    :LOCATE OF BEAM GUN
1140 '
1150 '== COLD START ==
1160 WIDTH 40,25:CONSOLE 0,25,0,1                    'SET TV MODE
1170 PRINT CHR$(12);                                 'CLEAR
1180 BMAX=8:HSCR=0
1190 YBEAM=0
1200 XGUN=18:YGUN=20
1210 GOSUB 1320                                      'CALL PRINT GAME ARER
1220 GOSUB 1520                                      'CALL PRINT BEAM GUN
1230 GOSUB 1580                                      'KEYSCAN
1240 GOSUB 1800                                      'CALL MOVE BEAM
1250 GOTO 1230
1260 '
1270 '-------
1280 '  SUB
1290 '-------
1300 '
1310 '== PRINT GAME AREA ==
1320 COLOR 1
1330 PRINT "=======================================";
1340 COLOR 7
1350 PRINT "                                        ";
1360 PRINT "                                        ";
1370 PRINT "                                        ";
1380 COLOR 2
1390 PRINT "                                        ";
1400 COLOR 3
1410 PRINT "                                        ";
1420 COLOR 6:PRINT "      -- P A N I C --   ";
1430 COLOR 7:PRINT "by K.TSUKAGOSHI "
1440 COLOR 1
1450 PRINT "=======================================";
1460 LOCATE 0,22
1470 COLOR 5:PRINT "=======================================";
1480 COLOR 6:PRINT "                                       ";
1490 LINE (0,24)-(38,24),"■",6,BF
1500 '
1510 '== PRINT BEAM GUN ==
1520 COLOR 1:LOCATE XGUN,YGUN   :PRINT " ▲ ";
1530         LOCATE XGUN,YGUN+1:PRINT " ■ ";
1540 COLOR 5:LOCATE XGUN,YGUN+2:PRINT "   ";
1550 RETURN
1560 '
1570 '== KEY SCAN ==
1580 I0=INP(0):I9=INP(9)                             'KEYSCAN
1590 IF I0=255 AND I9=255 THEN RETURN                'NO TOUCH TEHN RET
1600 IF I0=239 THEN 1660                             'LEFT ?
1610 IF I0=191 THEN 1700                             'RIGHT ?
1620 IF I9=191 THEN 1740                             'SPACE ?
1630 RETURN                                          'OTHE KEY THEN RET
1640 '
1650 '== MOVE LEFT BEAM GUN ==
1660 IF XGUN<2 THEN RETURN                           'LEFT EDGE ?
1670 XGUN=XGUN-1:GOTO 1510
1680 '
1690 '== MOVE RIGHT BEAM GUN ==
1700 IF XGUN>34 THEN RETURN                          'RIGHT EDGE ?
```

```
1710 XGUN=XGUN+1:GOTO 1510
1720 '
1730 '== SHOOT BEAM GUN ==
1740 IF YBEAM<>0 THEN RETURN                     'EXIST BEAM
1750   XBEAM=XGUN+2:YBEAM=19                     'INITIAL (XBEAM,YBEAM)
1760   LOCATE XBEAM,YBEAM:PRINT "▌";             'SHOOT BEAM
1770   RETURN
1780 '
1790 '== MOVE BEAM ==
1800 IF YBEAM=0 THEN RETURN                      'NO BEAM
1810   LOCATE XBEAM,YBEAM:PRINT " ";             'ERASE BEAM
1820   YBEAM=YBEAM-2                             'BEAM UP
1830   IF YBEAM<BMAX THEN YBEAM=0:RETURN         'BEAM END
1840     LOCATE XBEAM,YBEAM:PRINT "▌";           'NEW BEAM
1850     RETURN
```

と決めるのです。したがって**ビームを発射できるのは，条件①が成立するときだけ**です。リストの1740行を見てください。

YBEAM<>0――――――②

では，すでに**ビームが存在しています**から，ビームは発射できません。したがってすぐに

RETURN

しています。

②でないとき，すなわち①のときは，

スペース・キー

の入力をキャッチし，もし押されていれば

(XBEAM, YBEAM)

の値を**ビームの初期値**にセットしてやります。

ビームのX座標＝ビーム砲＋２（第４―10図）
Y座標＝19（第４―９図）

としてやれば良いでしょう（1750～1760行）。これによりYBEAMの値は，自動的に②の条件を満たすことに注意してください。

以上をまとめると，ビーム発射ルーチンの役目は次の一つになります。

① YBEAMの値により，**ビームのある・なし**を判定する。

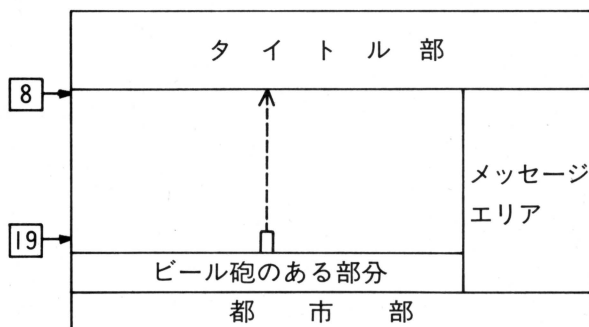
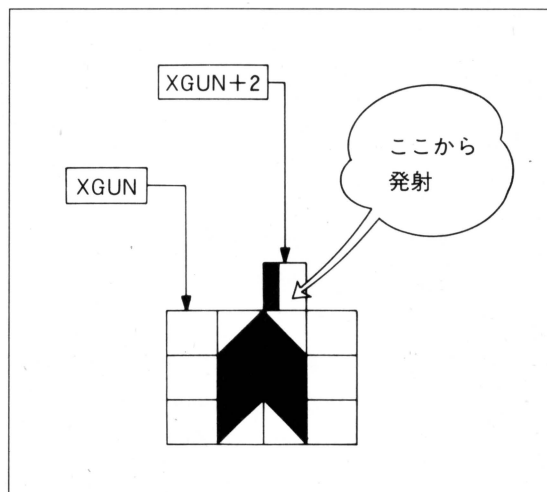
② **ビーム発射の条件**

YBEAM＝0
スペース・キーが押された

が満たされたら，ビームの初期値をセットする

ビーム移動ルーチンの製作

次は，**ビーム移動ルーチン**を考えます。

ここの基本的な考え方は，ビーム砲の移動と同じです。

《第４―９図》ビーム砲の動ける範囲

《第４―10図》ビーム砲の発射位置

ここで二つ程説明を加えておきます。

① 1820行を御覧ください。YBEAMは，ビームの

タテ座標です。普通なら

YBEAM＝YBEAM－1

とするところを

YBEAM＝YBEAM－2

としているのに注意してください。これは，ビームを一度に

2キャラクタ分進めている

ことになります。すなわち

ビームは通常の**2倍のスピード**

で動くことになります。BASICのノロさをカバーする秘密のテクニックです。あなたも大いに活用しましょう。実際**リスト4—10**をRUNさせると，

マシン語並のスピード

でビームが上昇するのがわかります。

② ビームは，放っておくとやがて画面の上端に到達します。したがって，プログラムの中にそのチェックを入れる必要があります。その手順は，次の通りです。

- **変数BMAXに，ビームの上限の位置**を定義してあります（1180行)。これは，タイトルの一段下の位置です。
- YBEAM<BMAX
となったときが，ビームが上限を越えたときです（1830行)。タテ座標の場合，数値が小さい方が上の位置になることに注意してください。
- ビームが上限に達した場合は，
YBEAM＝0
として，ビームの存在を消去します（1820行)。

リスト4—10の完成

以上の説明で**リスト4—10**は，すべて御理解いただけると思います。それでは，**リスト4—10**を走らせたところをお目にかけましょう。**写真8**です。今度は，

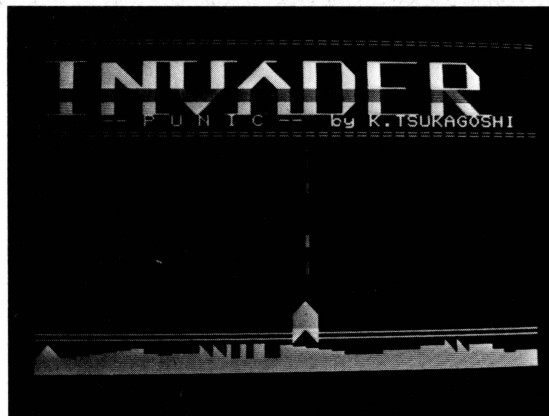
〝SPACE〞

の文字でなく，ちゃんと

ビーム

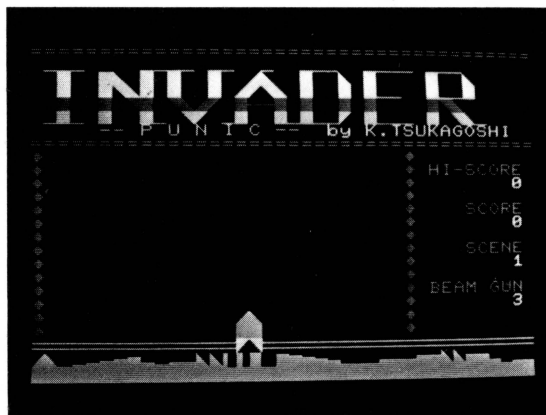
が出ていますね。その**高速性に御注目**ください。もちろん，ビーム砲も左右に動きますよ。

どうですか？ あなたのマシンでもここまでできましたか？ うまくいきましたか？ いやあ，感激でしょう。高いマイコン教室に通っても，なかなかここまでは教えてくれませんよ。いやあ，

GAMINGへの招待

《写真8》高速ビーム！

って，本当にいいですね！——ミズノ・ハルオの弟子。

編三、登場！

さて，次に**写真9**を御覧ください。これは，**リスト4—11**を走らせたものです。**写真8**と，どこが違っていますか？

《写真9》メッセージ・エリアの追加

さあ，次の挑戦は，

メッセージ・エリア

の追加です。これを入れると，画面が益々GAME画面らしくなってきます。表示は，

HI－SCORE：最高点
SCORE：現在の得点
SCENE：面の数
BEAM GUN：残りビーム砲の数

の四つとしました。これだけあれば，十分でしょう。もっともまだ**GAMEの構想**すらできていないのに，この先だいじょうぶでしょうね？

リスト4—11で追加・変更した部分は，とくに難し

```
1000 '==============================
1010 '   INVADER PANIC *--list 11--
1020 '       1982.5.4-?.??
1030 '             by K.TSUKAGOSHI
1040 '==============================
1050 '
1060 '--------
1070 '   MAIN
1080 '--------
1090 '
1100 'I0,I9          :VALUE OF INP
1110 'BMAX           :MAX OF YBEAM
1120 'XBEAM,YBEAM    :LOCATE OF BEAM
1130 'XGUN ,YGUN     :LOCATE OF BEAM GUN
1140 'HSCR           :HI-SCORE
1150 'SCR            :SCORE
1160 'SN             :SCENE
1170 'LBGUN          :LEFT OF BEAM GUN
1180 '
1190 '== COLD START ==
1200 WIDTH 40,25:CONSOLE 0,25,0,1               'SET TV MODE
1210 PRINT CHR$(12);                            'CLEAR
1220 BMAX=8:HSCR=0
1230 '
1240 '== HOT START ==
1250 YBEAM=0
1260 XGUN=18:YGUN=20
1270 SCR=0:LBGUN=3:SN=1
1280 GOSUB 1390                                 'CALL PRINT GAME ARER
1290 GOSUB 1660                                 'CALL PRINT BEAM GUN
1300 GOSUB 1720                                 'KEYSCAN
1310 GOSUB 1940                                 'CALL MOVE BEAM
1320 GOTO 1300
1330 '
1340 '-------
1350 '   SUB
1360 '-------
1370 '
1380 '== PRINT GAME AREA ==
1390 COLOR 1
1400 PRINT "======================================";
1410 COLOR 7
1420 PRINT " ▁▁   ▁▁▁ ▁▁ ▁ ▁▁▁ ▁▁▁▁ ▁▁▁      ";
1430 PRINT "  ■ ▄■■ ■ ▄▄■ ▖■  ■▖        ";
1440 PRINT "  ■ ■▀■ ■ ■▀■ ■■▁  ■▁■      ";
1450 COLOR 2
1460 PRINT "  ■ ■▀■▀▄▀■■■ ■ ■■  ■▀▄     ";
1470 COLOR 3
1480 PRINT " ▁■▁■ ■▁▀▄▁■ ■■▁▀■▁ ■▀▄▁    ";
1490 COLOR 6:PRINT "      -- P A N I C --   ";
1500 COLOR 7:PRINT "by K.TSUKAGOSHI "
1510 COLOR 1
1520 PRINT "======================================";
1530 LOCATE 0,22
1540 COLOR 5:PRINT "════════════════════════════════════";
1550 COLOR 6:PRINT "▲▁▁▄▄▁▁▄▌▌■ ■▄▁▁▁▁▁▄▄▄▄▁ ▁▁";
1560 LINE (0,24)-(38,24),"■",6,BF
1570 COLOR 2:FOR Y=8 TO 21
1580             LOCATE 0,Y:PRINT "◆";:LOCATE 28,Y:PRINT "◆";
1590          NEXT
1600 COLOR 4:LOCATE 30, 9:PRINT "HI-SCORE";:GOSUB 2020
1610 COLOR 4:LOCATE 30,12:PRINT "   SCORE";:GOSUB 2070
1620 COLOR 4:LOCATE 30,15:PRINT "   SCENE";:GOSUB 2120
1630 COLOR 4:LOCATE 30,18:PRINT "BEAM GUN";:GOSUB 2170
1640 '
1650 '== PRINT BEAM GUN ==
1660 COLOR 1:LOCATE XGUN,YGUN  :PRINT " ▲ ";
1670          LOCATE XGUN,YGUN+1:PRINT " ■ ";
1680 COLOR 5:LOCATE XGUN,YGUN+2:PRINT "=▼=";
1690 RETURN
1700 '
```

```
1710 '== KEY SCAN ==
1720 I0=INP(0):I9=INP(9)                          'KEYSCAN
1730 IF I0=255 AND I9=255 THEN RETURN             'NO TOUCH TEHN RET
1740 IF I0=239 THEN 1800                          'LEFT ?
1750 IF I0=191 THEN 1840                          'RIGHT ?
1760 IF I9=191 THEN 1880                          'SPACE ?
1770 RETURN                                       'OTHE KEY THEN RET
1780 '
1790 '== MOVE LEFT BEAM GUN ==
1800 IF XGUN<2 THEN RETURN                        'LEFT EDGE ?
1810 XGUN=XGUN-1:GOTO 1650
1820 '
1830 '== MOVE RIGHT BEAM GUN ==
1840 IF XGUN>23 THEN RETURN                       'RIGHT EDGE ?
1850 XGUN=XGUN+1:GOTO 1650
1860 '.
1870 '== SHOOT BEAM GUN ==
1880 IF YBEAM<>0 THEN RETURN                      'EXIST BEAM
1890   XBEAM=XGUN+2:YBEAM=19                      'INITIAL (XBEAM,YBEAM)
1900   LOCATE XBEAM,YBEAM:PRINT "▮";              'SHOOT BEAM
1910   RETURN
1920 '
1930 '== MOVE BEAM ==
1940 IF YBEAM=0 THEN RETURN                       'NO BEAM
1950   LOCATE XBEAM,YBEAM:PRINT " ";              'ERASE BEAM
1960   YBEAM=YBEAM-2                              'BEAM UP
1970   IF YBEAM<BMAX THEN YBEAM=0:RETURN          'BEAM END
1980     LOCATE XBEAM,YBEAM:PRINT "▮";            'NEW BEAM
1990     RETURN
2000 '
2010 '== PRINT HI-SCORE ==
2020 COLOR 7
2030 LOCATE 33,10:PRINT USING "#####";HSCR;
2040 RETURN
2050 '
2060 '== PRINT SCORE ==
2070 COLOR 7
2080 LOCATE 33,13:PRINT USING "#####";SCR;
2090 RETURN
2100 '
2110 '== PRINT SCENE ==
2120 COLOR 7
2130 LOCATE 35,16:PRINT USING "###";SN;
2140 RETURN
2150 '
2160 '== PRINT LEFT OF BEAM GUN ==
2170 COLOR 7
2180 LOCATE 37,19:PRINT USING "#";LBGUN;
2190 RETURN
```

い部分はありませんから説明はいらないと思います。

これら四つの情報を表示する部分は，それぞれ

独立したサブルーチン

にしてあります。したがってGAME進行にしたがって，これらの値が変化したときは，それぞれ

最高点：ＨＳＣＲ

得点：ＳＣＲ

面：ＳＮ

ビーム砲の残り数：ＬＢＧＵＮ

のうち概当する**変数の値を変え**，

最高点：2010行

得点：2060行

面：2110行

ビーム砲の残り数：2160行

の**概当するサブルーチンをＣＡＬＬ**すれば良いのです。

リスト４―11で変更したもう一点は，

ビーム砲の移動範囲

です。これは，メッセージ・エリアを設けたため移動可能領域が，**右側で少し狭く**なってしまったからです。仕方ありませんね。変更箇所は，1840行の

ＸＧＵＮ＞23

のところです。

〈糾弾コーナー〉

？？：素朴な糾弾ですが——。

ツカ：ハイ，ハイ（ドキ・ドキ）。

？？：　**ビーム**

と　**ビーム砲**

の違いは何ですか？

ツカ：ああ，何だあ。イ，イヤ，とても良い御質問ですね。

編三：ハイ，ハイ，その質問なら，私が答えます！

ツカ：おや？　編三さん，珍しい。おひさしぶりですね。

J君：アレッ，編三君，いないと思ったらこんな所でサボッている。

編三：とんでもない。これからツカさんに代わって御質問にお答えしようとしているところですよ。人聞きの悪い。

J君：人相の悪い。イヤ，コラ，恥かくからやめとけ！

編三：なーに，泣く子もだまる鬼の〝マイコン・ポスト〟で鍛えたこの細腕。今年の初夢が良かったから大丈夫ですよ。

J君：大丈夫かな。

ツカ：大丈夫ですよ。

編三：**ビーム砲**は，ビームを撃つ砲。**ビーム**は，ビーム砲から発射されたインベーダー攻撃用のタマですよ。

ツカ：拍手！

J君：何だ，それだけ？

M子ちゃん：何だ，そんなことならあたしだって知ってるわ。

（一同）：オヤ，M子ちゃんおひさしぶり！

M子ちゃん：エヘヘヘヘ。

J君：しかし，日本も広いようで狭いですなあ。

編三：狭い日本，そんなに急いでどこへ行く？

集長：アホか。

休　　憩

さあ，**リスト4—11**まで来ましたよ。どうですか，ちゃんとここまで来られましたでしょうか？　難しい？　イヤ，イヤ，どうも。

まあ，とにかくここらで一服しましょうか？

我々のやっているのは，GAME作りです。だから何が何でも完璧にマスターしなければならない——という性質のものではありません。御一緒に楽しんでいたけれれば良いのです。あなたの

マイコンを遊び遊具

として，そこをBASEとして遊んでしまえば良いのです。もちろんマスターしていただければ，それに越したことはありませんが。

話を変えましょう。今製作進行中の

インベーダー・パニック

は，**各リスト**が

完全に独立

しています。それは，**それだけで一つのプログラムとして動く**わけです。ただリストがあとの方に行くほど

いろいろな機能が追加

されているだけです。もしあなたが**リスト4—4**まで理解できたなら，ひとまずそこまでのプログラムで楽しんでください。そしてあとは気楽な読み物として，気を張らずに読み流してください。なぜなら〝GAMINGへの招待〟は，

教科書，解説書

の類ではないのですから。やがてあなたのマイコン・ライフの中で，あなたの学習がさらに進んだ頃，今度は**リスト4—5**も理解できるかもしれません。

このように気楽に一歩ずつ進んでいき，やがてすべてのリストがマスターできたとき，あなたのマイコンは，

インベーダー・パニック

のGAMEマシンとして，あなたをむかえてくれることでしょう。

インベーダーの登場

次は，**リスト4—12**への挑戦です。今度は，何が飛び出すでしょうか？　さっそくプログラムを走らせてみましょう。

RUN↗

写真10が実行させたところです。やや遂に

インベーダー

が登場してきました。インベーダーが

左から右へ

無気味に動いています。ビームを動かし，ビームを撃ってみましょう。ビームは，**インベーダーの体を通過**していきますね？　インベーダーは死にません。

そうです。**リスト4—12**では，

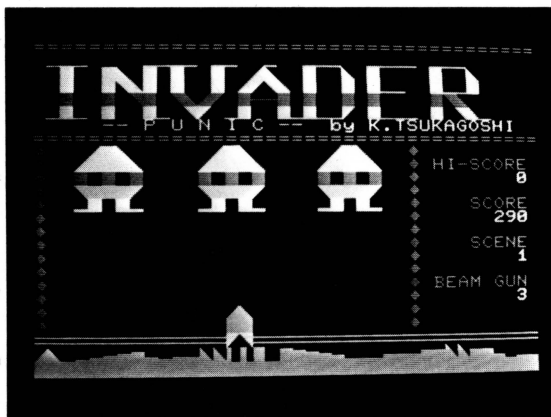
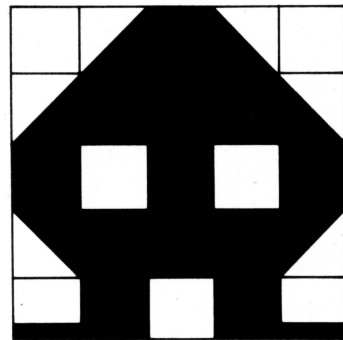
インベーダーを登場させ

インベーダーを動かす

だけが目標です。それにしても，画面右に消えたインベーダーがまた画面左から現われてきます。何だか難

```
1000 '============================
1010 '  INVADER PANIC --list 12--
1020 '      1982.5.4-?.??
1030 '             by K.TSUKAGOSHI
1040 '============================
1050 '
1060 '--------
1070 '  MAIN
1080 '--------
1090 '
1100 '── VARIABLE ──
1110 'IO,I9          :VALUE OF INP
1120 'BMAX           :MAX OF YBEAM
1130 'XBEAM,YBEAM    :LOCATE OF BEAM
1140 'XGUN ,YGUN     :LOCATE OF BEAM GUN
1150 'HSCR           :HI-SCORE
1160 'SCR            :SCORE
1170 'SN             :SCENE
1180 'LBGUN          :LEFT OF BEAM GUN
1190 '
1200 '── DIMENSION ──
1210 'IV$(4)         :INVADE
1220 '
1230 '== COLD START ==
1240 WIDTH 40,25:CONSOLE 0,25,0,1'                   SET TV MODE
1250 PRINT CHR$(12);'                                CLEAR
1260 BMAX=8:HSCR=0
1270 DIM I$(4),IV$(4):GOSUB 1740'                    INITIAL INVADER DATA
1280 '
1290 '== HOT START ==
1300 YBEAM=0:CLK=0
1310 XGUN=18:YGUN=20
1320 SCR=0:LBGUN=3:SN=1
1330 GOSUB 1450'                                     CALL PRINT GAME ARER
1340 GOSUB 1910'                                     CALL PRINT BEAM GUN
1350 GOSUB 1970'                                     KEYSCAN
1360 GOSUB 2190'                                     CALL MOVE BEAM
1370 IF INT(RND(1)*50)<10 THEN GOSUB 2280'           1/5 CALL MOVE INVADER
1380 GOTO 1350
1390 '
1400 '-------
1410 '  SUB
1420 '-------
1430 '
1440 '== PRINT GAME AREA ==
1450 COLOR 1
1460 PRINT "========================================";
1470 COLOR 7
1480 PRINT " ___  ___ ___ _ __ ___ ___          ";
1490 PRINT " ■ ■◣■■ ■◢■ ■◣■ ■ ◣          ";
1500 PRINT " ■ ■◥■■ ■◢■ ■■▁ ■▁■          ";
1510 COLOR 2
1520 PRINT " ■ ■◥◥◢ ■━■■ ■■ ■◣          ";
1530 COLOR 3
1540 PRINT " ▁■▁■ ■▁◥▁■ ■▁■◢■▁■ ■◥▁        ";
1550 COLOR 6:PRINT "      -- P A N I C --   ";
1560 COLOR 7:PRINT "by K.TSUKAGOSHI "
1570 COLOR 1
1580 PRINT "========================================";
1590 LOCATE 0,22
1600 COLOR 5:PRINT "════════════════════════════════════════";
1610 COLOR 6:PRINT "▲▁▁▁◢◣■■ ■■▁▁▁▁▁▁▁▁◢◣▁▁ ▁▁";
1620 LINE (0,24)-(38,24),"■",6,BF
1630 COLOR 2:FOR Y=8 TO 21
1640           LOCATE 0,Y:PRINT "◆";:LOCATE 28,Y:PRINT "◆";
1650         NEXT
1660 COLOR 4:LOCATE 30, 9:PRINT "HI-SCORE";:GOSUB 2340
1670 COLOR 4:LOCATE 30,12:PRINT "   SCORE";:GOSUB 2390
1680 COLOR 4:LOCATE 30,15:PRINT "   SCENE";:GOSUB 2440
1690 COLOR 4:LOCATE 30,18:PRINT "BEAM GUN";:GOSUB 2490
1700 GOSUB 1910'                                     PRINT BEAM GUN
```

```
1710 GOTO 1830'                                                  PRINT INVADER
1720 '
1730 '== INITIAL INVADER DATA ==
1740 RESTORE 2570
1750 FOR I=0 TO 4
1760   READ I$(I):IV$(I)=""
1770   FOR J=1 TO 3'                                             IV$=I$*3
1780     IV$(I)=IV$(I)+I$(I)
1790 NEXT J,I
1800 RETURN
1810 '
1820 '== PRINT INVADER ==
1830 COLOR 7:LOCATE 1,8 :PRINT IV$(0);
1840 COLOR 7:LOCATE 1,9 :PRINT IV$(1);
1850 COLOR 2:LOCATE 1,10:PRINT IV$(2);
1860 COLOR 3:LOCATE 1,11:PRINT IV$(3);
1870 COLOR 7:LOCATE 1,12:PRINT IV$(4);
1880 RETURN
1890 '
1900 '== PRINT BEAM GUN ==
1910 COLOR 1:LOCATE XGUN,YGUN   :PRINT " ▲ ";
1920         LOCATE XGUN,YGUN+1:PRINT " ■ ";
1930 COLOR 5:LOCATE XGUN,YGUN+2:PRINT "=◢◣=";
1940 RETURN
1950 '
1960 '== KEY SCAN ==
1970 I0=INP(0):I9=INP(9)'                                        KEYSCAN
1980 IF I0=255 AND I9=255 THEN RETURN'                           NO TOUCH TEHN RET
1990 IF I0=239 THEN 2050'                                        LEFT ?
2000 IF I0=191 THEN 2090'                                        RIGHT ?
2010 IF I9=191 THEN 2130'                                        SPACE ?
2020 RETURN'                                                     OTHE KEY THEN RET
2030 '
2040 '== MOVE LEFT BEAM GUN ==
2050 IF XGUN<2 THEN RETURN'                                      LEFT EDGE ?
2060 XGUN=XGUN-1:GOTO 1900
2070 '
2080 '== MOVE RIGHT BEAM GUN ==
2090 IF XGUN>23 THEN RETURN'                                     RIGHT EDGE ?
2100 XGUN=XGUN+1:GOTO 1900
2110 '
2120 '== SHOOT BEAM GUN ==
2130 IF YBEAM<>0 THEN RETURN'                                    EXIST BEAM
2140   XBEAM=XGUN+2:YBEAM=19'                                    INITIAL (XBEAM,YBEAM)
2150   LOCATE XBEAM,YBEAM:PRINT "▮ ";'                           SHOOT BEAM
2160   RETURN
2170 '
2180 '== MOVE BEAM ==
2190 IF YBEAM=0 THEN RETURN'                                     NO BEAM
2200   COLOR 6
2210   LOCATE XBEAM,YBEAM:PRINT "  ";'                           ERASE BEAM
2220   YBEAM=YBEAM-2'                                            BEAM UP
2230   IF YBEAM<BMAX THEN YBEAM=0:RETURN'                        BEAM END
2240     LOCATE XBEAM,YBEAM:PRINT "▮ ";'                         NEW BEAM
2250     RETURN
2260 '
2270 '== MOVE INVADER ==
2280 FOR I=0 TO 4
2290   IV$(I)=RIGHT$(IV$(I),2)+LEFT$(IV$(I),25)
2300 NEXT
2310 GOTO 1830
2320 '
2330 '== PRINT HI-SCORE ==
2340 COLOR 7
2350 LOCATE 33,10:PRINT USING "#####";HSCR;
2360 RETURN
2370 '
2380 '== PRINT SCORE ==
2390 COLOR 7
2400 LOCATE 33,13:PRINT USING "#####";SCR;
```

```
2410 RETURN
2420 '
2430 '== PRINT SCENE ==
2440 COLOR 7
2450 LOCATE 35,16:PRINT USING "###";SN;
2460 RETURN
2470 '
2480 '== PRINT LEFT OF BEAM GUN ==
2490 COLOR 7
2500 LOCATE 37,19:PRINT USING "#";LBGUN;
2510 RETURN
2520 '
2530 '--------
2540 '   DATA
2550 '--------
2560 '
2570 DATA " ◢█◣      ":'                           INVADER DATA
2580 DATA "◢███◣     "
2590 DATA "█ █ █     "
2600 DATA "◥███◤     "
2610 DATA "_█ █_     "
```

《写真10》インベーダー登場

《第4—11図》インベーダーの設計

しそうですね。

〈糾弾コーナー〉

？？：とてもインベーダーには見えません。

ツカ：スイマセン。

インベーダー表示の準備

それでは，順に製作していきましょう。まずインベーダーのキャラクタです。私の場合，**第4—11図**のようにしました。自分の好み，システムに合わせて変更してください。

このインベーダーのデータが，**リスト4—12**の2570行～2610行に**DATA文**として収めてあります。大きさは，

5×5

です。

次にインベーダーの**移動可能範囲**を見ますと，**第4—12図**のように

27キャラクタ分

ありますから，**第4—13図**のように

3匹のインベーダーを配置

することにします。1730行～1800行を御覧ください。これは**第4—13図**のフォーマットにしたがってインベーダーＤＡＴＡを

ＩＶ＄（0）：1行目

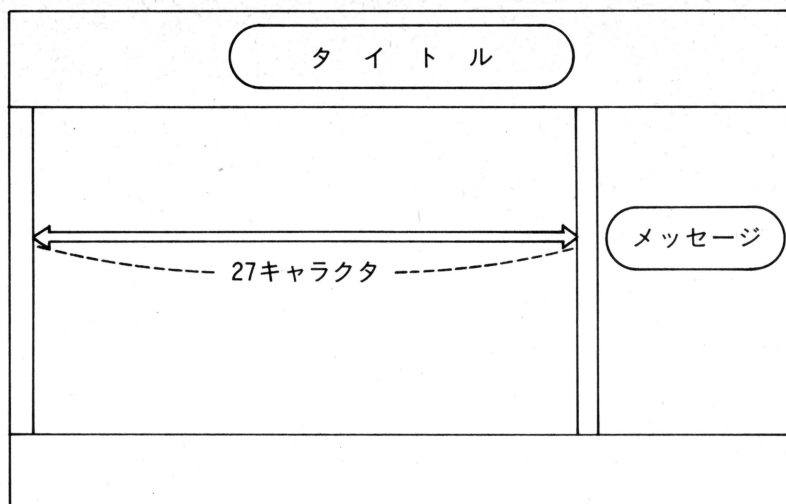
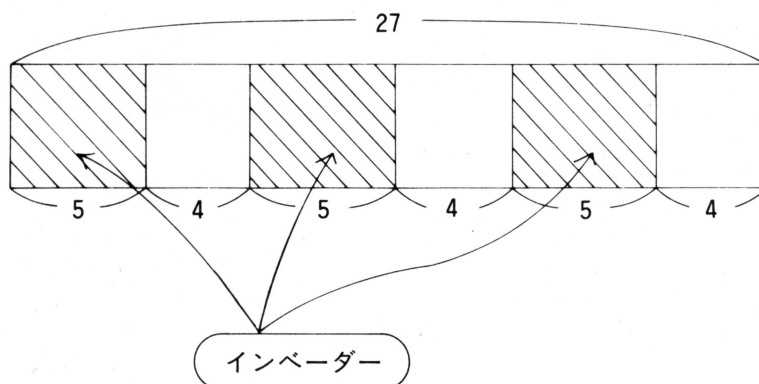
ＩＶ＄（1）：2行目

ＩＶ＄（2）：3行目

ＩＶ＄（3）：4行目

ＩＶ＄（4）：5行目

のように代入しているのです。ここでは，ＲＥＡＤ文を用いていますが，わかりにくい人は

ＩＶ＄（0）＝〝――3匹分のデータ――〞

《第４—12図》インベーダーの移動可能範囲

《第４—13図》３匹のインベーダーを配置

ＩＶ＄（１）＝〝――３匹分のデータ――〃

ＩＶ＄（２）＝〝――３匹分のデータ――〃

ＩＶ＄（３）＝〝――３匹分のデータ――〃

ＩＶ＄（４）＝〝――３匹分のデータ――〃

のように直接代入すると良いでしょう。

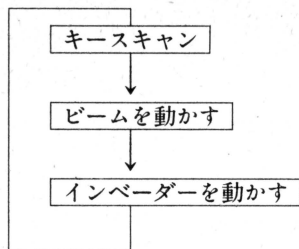
以上で**３匹のインベーダーのＤＡＴＡが，配列ＩＶ＄に用意された**ことになります。すると，1820～1880行のＰＲＩＮＴ文で画面に表示されるのがわかるでしょう。そこで今みたサブルーチンをまとめると，次のようになります。

1730～1800：インベーダーのＤＡＴＡの準備
1820～1880：インベーダーの表示

インベーダー移動の流れは

以上の予備知識をもとに，メインルーチンの流れを見てみましょう。

まず1270行でインベーダーのＤＡＴＡを用意します。次に1330行でＧＡＭＥの画面を表示しますが，この中でインベーダーも表示しています，(1710行を見てください)。

1350行～1380行が，実際にＧＡＭＥが進行する部分です。この部分は，

のようにはっています。

2280～2310：インベーダーを動かす

ルーチンは，1回CALLされるたびに，

インベーダーを右に2キャラクタ分

動かします。以上がメインルーチンの流れです。

そこで最後に2280行からの**インベーダーを動かすルーチン**を見てみましょう。

2290行の

LEFT$

RIGHT$

は何をしているのかわかりますか？　これは，**第4―14図**のように配列IV$に収められているインベーダーのDATAを**左に回転**させているのです。(**RIGHT$，LEFT$の使い方**については，御自分のマニュアルの**文字関数**のところを見てくださいね)。こうしてデータを回転させたあと，インベーダーを表示してやれば，インベーダーが動くというわけです。

〈糾弾コーナー〉

？？：1370行の

IF　INT（RND～）

は何をしているのですか？

ツカ：ハハア，やはり気がつきましたか。この部分は，少し難しいのでわざと避けていたのですが，やはり設明しないといけないでしょうね。

？？：「マイコン誌」の精神に反します。

ツカ：1370行は，他の行と同じく単に

1370　GOSUB　2280

でも良いのです。ただここでは，

インベーダーの動きを不規則に

してGAMEを面白くするため，乱数を用いています。その使い方は御存知のように

0＜RND（1）＜1

の乱数が発生しますから

0＜RND（1）＊50＜50

となり，INTで小数点以下を切り捨てると，

INT（RND（1）＊50）は，

0，1，2，……，49

の50個のいずれかの値になります。この値が10より小になるということは，結局

5分の1の確率

で2280行のサブルーチンがCALLされることになります。これを画面で見ると，インベーダーの動きが**速かったり遅かったり不規則**になることになります。おわかりになったでしょうか？

？？：――

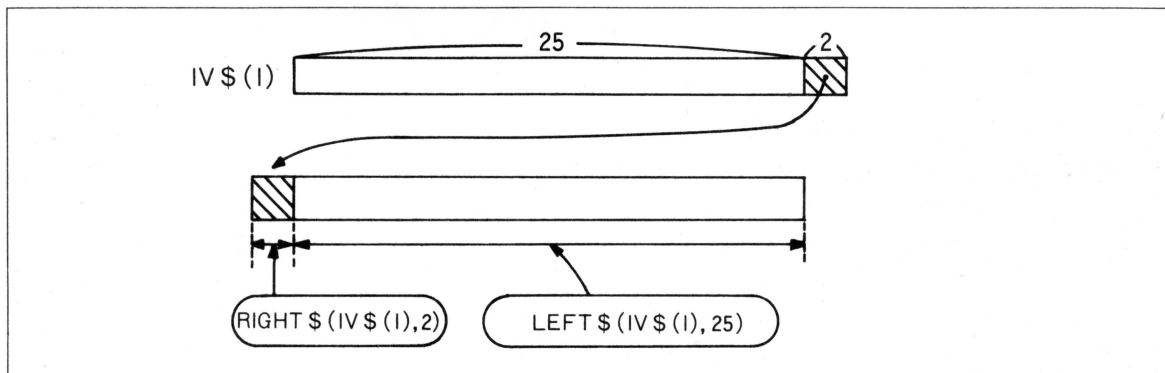
ツカ：そういえば，編三さんが現われませんね。

第3章のおわりに

第3章では**ビール砲を実際に発射**するところから始まって，**インベーダーを登場**させるところまでもっていきました。いわば双方の役者がそろったことになるわけです。さあ，今や地球をめぐる

攻　防　戦

が始まらんとしています。これから一体どんなゲームが展開されるのでしょうか？　はたまたゲーム作りは，**途中で挫折**するのでしょうか？　そして編三君の運命やいかに？

これから我々は，舞台を第4章に移し，その展開を見ていくことに致しましょう。

《第4―14図》インベーダーのデータの回転

「インベーダー・パニック」に挑戦
GAMEを完成させる

はじめに

第3章末期に，その醜い姿を地球上空に現わしたインベーダーの編隊は，地球侵略への野望をかけ，偵察活動を開始した。今や地球への攻撃は，時間の問題とされた。その

異星物侵略者（インベーダー）

をむかえうつ地球側は，超高性能ビーム砲を擁し，すでに防衛の準備は完了した。そして時代は，第4章へと移り，と同時にインベーダーはその先頭部隊として

ジュニア・インベーダー

を登場させてきた。この得体のしれない異星物は何か？　どんな攻撃をしかけてくるのか？　第4章，はじまり，はじまり。

ルールの暫定案

これから挑戦する**リスト4—13**，さっそく走らせてみましょう。

RUN↗

写真11が，走らせた直後のものです。ややっ，インベーダーの下に何やら怪しげな異星物が登場しましたね。これが，

ジュニア・インベーダー

です。彼等は，インベーダーとは逆に，左に移動しています。ためしにビーム砲で攻撃してみましょう。

[スペース] キー

を押してみます。ビームが，インベーダーに当たっても，ビームはインベーダーの体をつきぬけるだけで，インベーダーは死にません。しかし，ビームがジュニア・インベーダーに当たると，

ジュニア・インベーダが消え，
得点が加算

されることがわかります（**写真12**）。一体どうなっているのでしょうね？

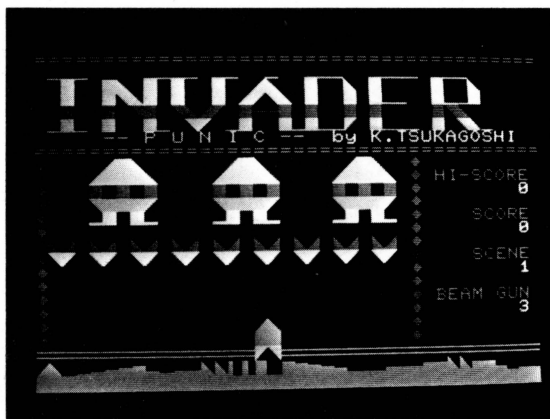
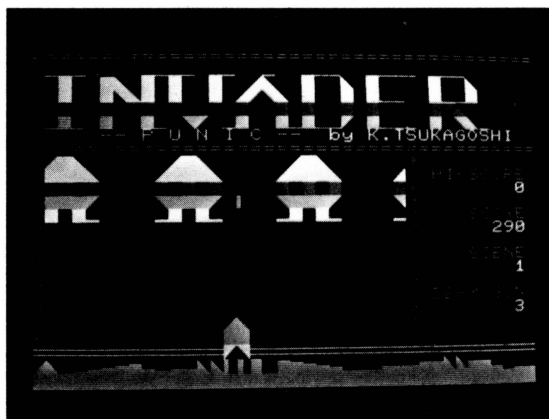
《リスト4—13》ジュニア・インベーダー登場リスト

```
1000 '=============================
1010 '   INVADER PANIC --list 13--
1020 '       1982.5.4-?.??
1030 '             by K.TSUKAGOSHI
1040 '=============================
1050 '
1060 '---------
1070 '   MAIN
1080 '---------
1090 '
1100 '--- VARIABLE ---
1110 'IO,I9          :VALUE OF INP
1120 'BMAX           :MAX OF YBEAM
```

```
1130 'XBEAM,YBEAM  :LOCATE OF BEAM (IF YBEAM=0 TEHN NO BEAM)
1140 'XGUN ,YGUN   :LOCATE OF BEAM GUN
1150 'XIV$         :LOCATE X OF INVADER
1160 'XJIV$        :LOCATE X OF JUNIOUR INVADER
1170 'LIV          :LEFT OF INVADER
1180 'LJIV         :LEFT OF JUNIOUR INVADER
1190 'ABLE         :IF ABLE=0 TEHN IT IS IMPOSSIBLE TO ATTACK INVADER.
1200 'HSCR         :HI-SCORE
1210 'SCR          :SCORE
1220 'SN           :SCENE
1230 'LBGUN        :LEFT OF BEAM GUN
1240 'CLK          :CLOCK COUNTER
1250 '
1260 '—— DIMENSION ——
1270 'IV$(4)       :INVADE
1280 'JI$(1)       :JUNIOR INVADER
1290 '
1300 '== COLD START ==
1310 WIDTH 40,25:CONSOLE 0,25,0,1'                  SET TV MODE
1320 PRINT CHR$(12);'                               CLEAR
1330 BMAX=8:HSCR=0
1340 DIM I$(4),IV$(4)
1350 '
1360 '== HOT START ==
1370 YBEAM=0:CLK=0
1380 XGUN=18:YGUN=20
1390 SCR=0:LBGUN=3:SN=1
1400 GOSUB 1900'                                    RESET INVADER DATA
1410 GOSUB 2000'                                    RESET JUNIOUR INVADER DATA
1420 GOSUB 1600'                                    CALL PRINT GAME ARER
1430 GOSUB 2220'                                    CALL PRINT BEAM GUN
1440 '
1450 '== GAME ==
1460   GOSUB 2280'                                  KEYSCAN
1470   GOSUB 2500'                                  CALL MOVE BEAM
1480   IF INT(RND(1)*50)<10 THEN GOSUB 2650'        1/5 CALL MOVE INVADER
1490   IF CLK=0 THEN GOSUB 2730'                    CALL MOVE JUNIOUR INVADER
1500   CLK=CLK+1:IF CLK=3 THEN CLK=0'               COUNTUP CLOCL
1510   IF YBEAM=15 THEN GOSUB 2800'                 CHECK JIV
1520   IF BMAX<YBEAM AND YBEAM <13 THEN GOSUB 2940' CHECK IV
1530 GOTO 1460
1540 '
1550 '-------
1560 '  SUB
1570 '-------
1580 '
1590 '== PRINT GAME AREA ==
1600 COLOR 1
1610 PRINT "=======================================";
1620 COLOR 7
1630 PRINT " ___  ____ __ _ ___ ____ ___     ";
1640 PRINT " ■ ▶■■ ■▲■ ■▶■  ■▶   ";
1650 PRINT " ■ ■▶■■ ■▲■ ■■▬ ■■■  ";
1660 COLOR 2
1670 PRINT " ■ ■▼▼■▬■■ ■■  ■▶    ";
1680 COLOR 3
1690 PRINT " ▬■ ■ ■▬ ▼ ■ ■■▬■▬ ■▶▬   ";
1700 COLOR 6:PRINT "    -- P A N I C --  ";
1710 COLOR 7:PRINT "by K.TSUKAGOSHI "
1720 COLOR 1
1730 PRINT "=======================================";
1740 LOCATE 0,22
1750 COLOR 5:PRINT "══════════════════════════════════════";
1760 COLOR 6:PRINT "▲▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬ ▬▬";
1770 LINE (0,24)-(38,24),"■",6,BF
1780 COLOR 2:FOR Y=8 TO 21
1790           LOCATE 0,Y:PRINT "◆";:LOCATE 28,Y:PRINT "◆";
1800         NEXT
1810 COLOR 4:LOCATE 30, 9:PRINT "HI-SCORE";:GOSUB 3100
1820 COLOR 4:LOCATE 30,12:PRINT "   SCORE";:GOSUB 3150
1830 COLOR 4:LOCATE 30,15:PRINT "   SCENE";:GOSUB 3200
1840 COLOR 4:LOCATE 30,18:PRINT "BEAM GUN";:GOSUB 3250
1850 GOSUB 2220'                                    PRINT BEAM GUN
```

```
1860 GOSUB 2090'                                    PRINT INVADER
1870 GOTO  2170'                                    PRINT JUNIOUR INVADER
1880 '
1890 '== INITIAL INVADER DATA ==
1900 RESTORE 3330
1910 FOR I=0 TO 4
1920   READ I$(I):IV$(I)=""
1930   FOR J=1 TO 3'                                IV$=I$*3
1940     IV$(I)=IV$(I)+I$(I)
1950 NEXT J,I
1960 XIV$ ="112110000112110000112110000"'           RESET XIV$
1970 LIV=3:RETURN
1980 '
1990 '== INITIAL JUNIOUR INVADER DATA ==
2000 JI$(0)="":JI$(1)=""'                           RESET J$( )
2010 FOR I=1 TO 9
2020   JI$(0)=JI$(0)+"◣◢ "
2030   JI$(1)=JI$(1)+"▼ "
2040 NEXT
2050 XJIV$="110110110110110110110110110"'           RESET XJIV$
2060 LJIV=9:ABLE=0:RETURN
2070 '
2080 '== PRINT INVADER ==
2090 COLOR 7:LOCATE 1,8 :PRINT IV$(0);
2100 COLOR 7:LOCATE 1,9 :PRINT IV$(1);
2110 COLOR 2:LOCATE 1,10:PRINT IV$(2);
2120 COLOR 3:LOCATE 1,11:PRINT IV$(3);
2130 COLOR 7:LOCATE 1,12:PRINT IV$(4);
2140 RETURN
2150 '
2160 '== PRINT JUNIOR INVADER ==
2170 COLOR 2:LOCATE 1,14:PRINT JI$(0);
2180 COLOR 7:LOCATE 1,15:PRINT JI$(1);
2190 RETURN
2200 '
2210 '== PRINT BEAM GUN ==
2220 COLOR 1:LOCATE XGUN,YGUN   :PRINT " ▲ ";
2230         LOCATE XGUN,YGUN+1:PRINT " ■ ";
2240 COLOR 5:LOCATE XGUN,YGUN+2:PRINT "═◢◣═";
2250 RETURN
2260 '
2270 '== KEY SCAN ==
2280 I0=INP(0):I9=INP(9)'                           KEYSCAN
2290 IF I0=255 AND I9=255 THEN RETURN'              NO TOUCH TEHN RET
2300 IF I0=239 THEN 2360'                           LEFT ?
2310 IF I0=191 THEN 2400'                           RIGHT ?
2320 IF I9=191 THEN 2440'                           SPACE ?
2330 RETURN'                                        OTHE KEY THEN RET
2340 '
2350 '== MOVE LEFT BEAM GUN ==
2360 IF XGUN<2 THEN RETURN'                         LEFT EDGE ?
2370 XGUN=XGUN-1:GOTO 2210
2380 '
2390 '== MOVE RIGHT BEAM GUN ==
2400 IF XGUN>23 THEN RETURN'                        RIGHT EDGE ?
2410 XGUN=XGUN+1:GOTO 2210
2420 '
2430 '== SHOOT BEAM GUN ==
2440 IF YBEAM<>0 THEN RETURN'                       EXIST BEAM
2450   XBEAM=XGUN+2:YBEAM=19'                       INITIAL (XBEAM,YBEAM)
2460   LOCATE XBEAM,YBEAM:PRINT "▮";'               SHOOT BEAM
2470   RETURN
2480 '
2490 '== MOVE BEAM ==
2500 IF YBEAM=0 THEN RETURN'                        NO BEAM
2510   COLOR 6
2520   LOCATE XBEAM,YBEAM:PRINT " ";'               ERASE BEAM
2530   YBEAM=YBEAM-2'                               BEAM UP
2540   IF YBEAM<BMAX THEN 2590'                     ERASE BEAM
2550     LOCATE XBEAM,YBEAM:PRINT "▮";'             NEW BEAM
2560     RETURN
2570 '
2580 '== ERASE BEAM ==
```

```
2590 YBEAM=0:IF ABLE=0 THEN RETURN'             BEAM END
2600 ABLE=ABLE-1'                                COUNT DOWN ABLE
2610 IF ABLE=0 THEN GOSUB 2000'                  RESURRECDTIN JUNIOUR INBVADER
2620   RETURN
2630 '
2640 '== MOVE INVADER ==
2650 FOR I=0 TO 4
2660   IV$(I)=RIGHT$(IV$(I),2)+LEFT$(IV$(I),25)
2670 NEXT
2680 GOSUB 2090'                                 PRINT INVADER
2690 XIV$=RIGHT$(XIV$,2)+LEFT$(XIV$,25)'         LOCATE X
2700 RETURN
2710 '
2720 '== MOVE JUNIOUR INVADER ==
2730 JI$(0)=RIGHT$(JI$(0),26)+LEFT$(JI$(0),1)
2740 JI$(1)=RIGHT$(JI$(1),26)+LEFT$(JI$(1),1)
2750 GOSUB 2170'                                 PRINT JUNIOUR INVADER
2760 XJIV$=RIGHT$(XJIV$,26)+LEFT$(XJIV$,1)'      LOCATE XJUNIOURINVADER
2770 RETURN
2780 '
2790 '== CHECK CRUSH OF JUNIOUR INVADER ==
2800 IF MID$(XJIV$,XBEAM,1)="0" THEN RETURN'     MISS !
2810   I=XBEAM'                                  SEARCH FOR LEFT OF 1
2820     IF MID$(XJIV$,I-1,1)="1" THEN I=I-1
2830   MID$(JI$(0),I)="  "'                      DELETE JUNIOUR INVADER
2840   MID$(JI$(1),I)="  "
2850   MID$(XJIV$,I)="00"
2860   GOSUB 2590'                               ERASE BEAM
2870   SCR=SCR+10:GOSUB 3150'                    SCORE COUNT UP
2880   LJIV=LJIV-1:IF LJIV>0 THEN RETURN'
2890     ABLE=5'                                 ATTACK INVADER OK !
2900     SCR=SCR+INT(RND(0)*3+1)*100:GOSUB 3150'
2910     RETURN
2920 '
2930 '== CHECK CRUSH OF INVADER ==
2940 IF ABLE=0 THEN RETURN'                      IMPOSSIBLE
2950 IF MID$(XIV$,XBEAM,1)<>"2" THEN RETURN'     MISS !
2960   I=XBEAM'                                  SEARCH FOR LEFT OF 1
2970     IF MID$(XIV$,I-1,1)="0" THEN 2980 ELSE I=I-1:GOTO 2970
2980   FOR J=0 TO 4
2990     MID$(IV$(J),I)="     "
3000   NEXT
3010   MID$(XIV$,I)="00000"
3020   SCR=SCR+100:GOSUB 3150'                   SCORE COUNT UP
3030   LIV=LIV-1:IF LIV>0 THEN RETURN'           EXIST INVADER
3040     SCR=SCR+1000:GOSUB 3150'                SCORE COUNT UP
3050     SN=SN+1:GOSUB 3200'                     SCENE COUNT UP
3060     GOSUB 1900:GOSUB 2000'                  RESET IV,JIV DATA
3070     RETURN
3080 '
3090 '== PRINT HI-SCORE ==
3100 COLOR 7
3110 LOCATE 33,10:PRINT USING "#####";HSCR;
3120 RETURN
3130 '
3140 '== PRINT SCORE ==
3150 COLOR 7
3160 LOCATE 33,13:PRINT USING "#####";SCR;
3170 RETURN
3180 '
3190 '== PRINT SCENE ==
3200 COLOR 7
3210 LOCATE 35,16:PRINT USING "###";SN;
3220 RETURN
3230 '
3240 '== PRINT LEFT OF BEAM GUN ==
3250 COLOR 7
3260 LOCATE 37,19:PRINT USING "#";LBGUN;
3270 RETURN
3280 '
```

```
3290 '--------
3300 '   DATA
3310 '---------
3320 '
3330 DATA " ▲      ":'                     INVADER DATA
3340 DATA "■■■     "
3350 DATA "■ ■ ■   "
3360 DATA "▼▼▼     "
3370 DATA "▄█ █▄   "
```

《写真11》ジュニアインベーダー登場

《写真12》消えたジュニアインベーダー

ここで"インベーダー・パニック"のルールがだいぶハッキリしてきましたので，お知らせしておきましょう。

〈ルール（暫定案）〉

① **ＧＡＭＥの目的**は，

インベーダー

ジュニア・インベーダー

に攻撃を加え，できるだけ高得点を取ること。

② ジュニア・インベーダーは，いつでも攻撃可能で，1匹攻撃するごとに10点加算される。

③ **9匹のジュニア・インベーダーをすべて消すと，初めてインベーダー攻撃可能**となる。このとき**ボーナス**として

１００点，２００点，３００点

のいずれかが得点に加算される。

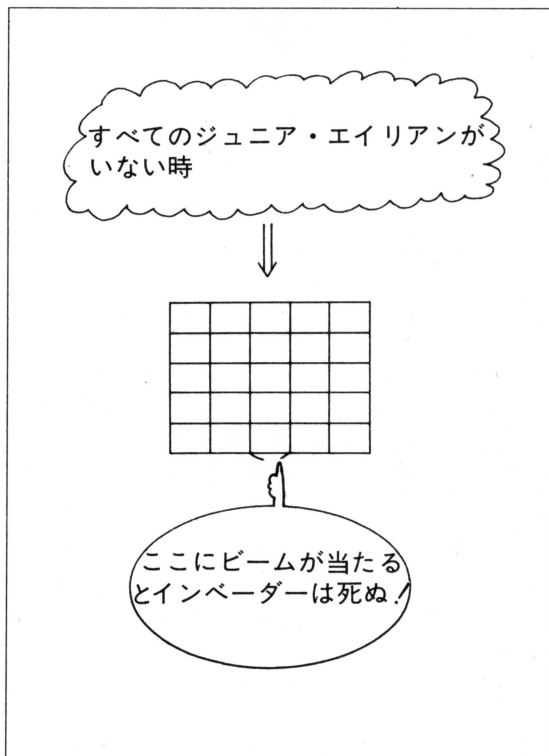
④ インベーダー攻撃可能な状態のとき，ビームが**インベーダーのちょうど真中**に当たるとインベーダーは死ぬ。その他の場所に当たっても死にませんから，御注意ください。(**第4—15図**)。インベーダーの得点は，100点です。

⑤ インベーダー攻撃可能な時間は，

ビーム5発

までです。ビーム5発を撃ち終わると，またジュニア・インベーダーが発生します。

⑥ インベーダー3匹全部を攻撃すると，ボーナスとして1000点が加算され，**1面の終了**となります。

以上がルールの暫定案です。ここに示した得点類は，あとで変更するたもしれません。**リスト4—13**は，ここまで遊べるように作ってあります。実際にプログラムも走らせて確かめてください。

《第4—15図》インベーダーが死ぬ時

ジュニア・インベーダーの処理

これからその**リスト4―13**を調べていくことになります。**リスト4―12**に比べ，だいぶ追加されています。どれもそれ程難しくないのですが，少しややこしいかもしれません。頭の混乱をきたさないように，注意深く読んでください。

最初は，ジュニア・インベーダーを動かすところから見てみましょう。これは，インベーダーを動かす原理とまったく同じですから，簡単に理解していただけることと思います。

まずジュニア・インベーダーのＤＡＴＡは，**第4―16図**のようになっています。これを９匹分，**第4―17図**のように配置します。そのＤＡＴＡを2010行～2040行で

ＪＩ＄（0）

ＪＩ＄（1）

《第4―16図》ジュニア・インベーダーのＤＡＴＡ

にセットします。データをセットしたあと，

2160～2190：ジュニア・インベーダーの表示

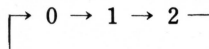
をＣＡＬＬすれば，ジュニア・インベーダーの編隊が表示されるわけです。

ジュニア・インベーダーを動かすのは，2720～2770行のサブルーチンです。インベーダーとは逆に，**データを右に回転**してやります。なおジュニア・インベーダーの場合，**1回に1キャラクタ分**しか動かいていません。

メインルーチンでは，このルーチンを1490行でＣＡＬＬしています。

変数ＣＬＫ

について説明を加えておきましょう。ＧＡＭＥ進行中プログラムは，1460～1530行の間をグルグルまわっています。そしてループするごとにＣＬＫは，

→ 0 → 1 → 2 ─┐（2から0へ戻る）

と値を変えていきます（1500行を見てください）。そしてちょうど

ＣＬＫ＝0

になったとき，このときだけ1490行のように2730行からの〝**ジュニア・インベーダー移動ルーチン**〟がＣＡＬＬされます。つまり，

ジュニア・インベーダーは

三回に一回しか動かない

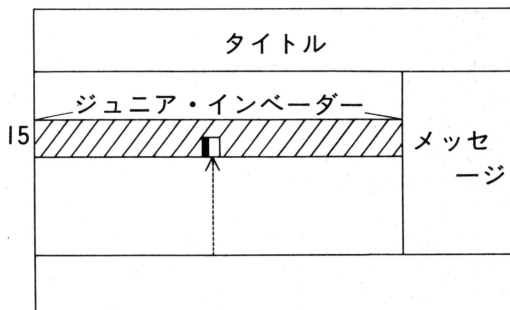
というわけです。以上のようなテクニックを用いれば，いろいろなものの**動くスピード**を変えることができます。

《第4―17図》ジュニア・インベーダーの配置

ジュニア・インベーダーを殺す

次に，ビームがジュニア・インベーダーに当たったかどうかを**チェックするルーチン**を考えてみます。それは，1510行でＣＡＬＬされている2800行からのサブルーチンです。

ビームが発射され，上昇していくと，やがて**ジュニア・インベーダーが存在する行**に達します(**第4―18図**)。それは，ちょうど

ＹＢＥＡＭ＝１５

《第4―18図》ビーム砲が上昇して

になったときです。これで1510行の意味は，おわかりになりましたね？

次は，どうして

ビームがジュニア・インベーダーに当たったか？

を判断するかです。**リスト4—13では，**

変数XJIV$

を導入し，この問題を解決しています。まずこの変数に2050行のように

0　と　1

の数をセットします。この数字の並びを良く観察してみてください。

110

の順に9回繰り返されているのがおわかりになるでしょう。つまりこの1と0は，

1：ジュニア・インベーダーがいる
0：ジュニア・インベーダーがいない

ことを示しているのです。

ジュニア・インベーダーは，GAME進行にしたがって動いていきます。2730行～2740行でジュニア・インベーダーのDATAを動かしていますね？　したがって

XJIV$

も同じように回転させてやります（2760行)。これで

XJIV$は配列JI$と同じ動き

をしていることがおわかりになったと思います。したがって

XIV$の左からXBEAM目

の値を調べれば，ジュニア・インベーダーの生死を判定できるわけです。

それでは，以上の予備知識をもとに，

2790～2910：ジュニア・インベーダー
生死チェック・ルーチン

を調べてみましょう。

まず2800行で

MID$（XJIV$，XBEAM，1）

で生死を判定するDATAを取り出します。もし**この値が0なら**

ビームははずれた！

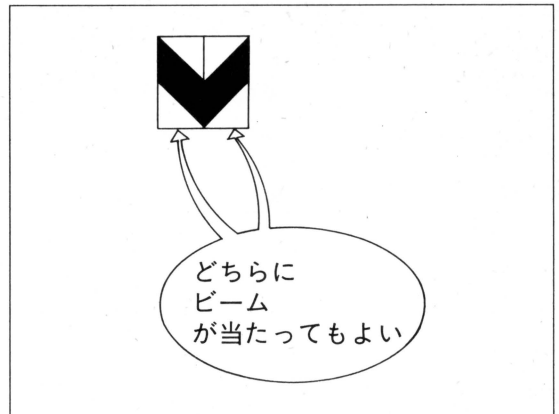
わけですから，何もしないでそのまま

RETURN

してやります。

ビームが当たった場合は，どうすれば良いでしょう？　次の処理が必要になります。

① **JI$，XJIV$から死んだジュニア・インベーダーのDATAを消す**

● ジュニア・インベーダーは，**ヨコに2キャラクタ分の大きさ**をもっています。したがってビームの当たった位置は，二種類の場合が考えられます**（第4—19図）**。

《第4—19図》ジュニア・インベーダーは横二つ分の大きさを持つ

● もし右側に当たった場合は，ヨコ座標を一つ左にずらしてやります。2820行はその調整を行っているところで，2810～2820行で

死んだジュニア・インベーダーの
左側の座標が変数Iに

求まります。

● **このIから右二つのDATA**を消してやれば良いのです。

JI$：右二つを空白に

XJIV$：右二つを0に

してやれば，ジュニア・インベーダーのDATAが消えます。

② **得点のカウント・アップ**

得点を表わす変数SCRを

＋10

してやり，新しい得点をメーセージ・エリアに表示します（2870行)。

③ **まだジュニア・インベーダーが残っているかチェックする**

ジュニア・インベーダーの数は

変数LJIV

に入っていて，初期値は9です（2060行)。したがってこのLJIVの値を

－1

してやります（2880行)。

④ **LJIV＝0になったら，インベーダー攻撃可能な状態にしてやる**

LJIV＝0

とは，すべてのジュニア・インベーダーが死んだことを意味します。そこでインベーダーの攻撃を可能な状態にしてやります。それには

変数ＡＢＬＥ

を用います。このＡＢＬＥには，二つの役割があります。

1）**インベーダーへの攻撃が可能かどうか**

ＡＢＬＥ＞０：攻撃可能
ＡＢＬＥ＝０：攻撃不可能

2）**インベーダーへの攻撃が可能なとき，残りのビームの数を表わす。**

暫定ルールで，ジュニア・インベーダーを全部消したあと，インベーダーへの攻撃が５回しかできなかったことを思い出してください。

ＡＢＬＥは，まず2060行で初期値０にセットされます。この状態では，**インベーダーへの攻撃ができません**。2930～3070行のサブルーチンで，インベーダーの生死を調べていますが，その入口の2940行で

ＡＢＬＥ＝０

だとすぐにＲＥＴＵＲＮしていることに注意してください。

さて以上のことをまとめると，

ＬＪＩＶ＝０

になったら

ＡＢＬＥ＝５

にしてやれば良いことがおわかりになったでしょう(2890行)。なお2900行は，すべてのジュニア・インベーダーが消えたとき，

ボーナスの得点

を加算しているところです。

インベーダーを殺す

最後にインベーダーを殺しましょう。

インベーダーの生死をチェックするルーチンは，

2930行～3070行

で，**ビームがインベーダーのところを通過**したとき，すなわちメインルーチンで

ＢＭＡＸ＜ＹＢＥＡＭ　ＡＮＤ　ＹＢＥＡＭ＜13

のときＣＡＬＬされます（1520行）。

それでは，そのＣＡＬＬされたサブルーチンを調べてみましょう。

① 攻撃不能（ＡＢＬＥ＝０：前述）なら，すぐに

ＲＥＴＵＲＮ

します（2940行）。

② インベーダーの存在を示す値は，

変数ＸＩＶ＄

に入っています。これは，1960行で定義しているように

１１２１１

でインベーダーを示しています。**真中の２がインベーダーの急所**です。したがって

ＭＩＤ＄（ＸＩＶ＄，ＸＢＥＡＭ，１）

の値が２なら，ビームがインベーダーに当たったことになります（2950）。

③ 2960行～2970行は，死んだインベーダーの左端の位置をＩに入れているところです。そこから右に五つ分のＤＡＴＡを消去します。

④ 得点は，100点を加算します（3020行）。

⑤ 変数ＬＩＶは，インベーダーの残り数で初期値は，３です（1970行）。これを

－１

します（3030行）。

⑥ もし，

ＬＩＶ＝０

になったら，

残りのインベーダー＝０

ですから，**１面終了の処理**を行います。

1）得点を最高の1000点加えます（3040行）。

2）面の数ＳＮを＋１して表示します（3050行）。

3）インベーダー，ジュニア・インベーダーの数を元に戻し，ＤＡＴＡを初期化します（3060行）。

以上でインベーダーの生死チェック・ルーチンはおしまいです。と同時に長い長い**リスト４―13**の解説が終了致しました。皆さん，お疲れさまでした。

〈糾弾コーナー〉

？？：すべてのジュニア・インベーダーを消したあと，インベーダーへの攻撃可能回数を

ＡＢＬＥ＝５

にセットしていますが，**いつカウント・ダウンしている**のですか？

ツカ：ＡＢＬＥのカウント・ダウンですね。これは，2930行～3070行の〝インベーダーの生死チェック・ルーチン〟では行っていません。ＡＢＬＥは，ビームを撃てる回数ですから，ビーム発射後**ビームが消滅するとき**カウント・ダウンしています。具体的には，2600行です。

？？：なぜ〝インベーダーの生死チェック・ルーチンで行わなかったのですか？

ツカ：このルーチンは，ビームがインベーダーの存在領域を通過中CALLされますから，一つのビームでも**複数回カウント・ダウン**されてしまいます。つまり5回撃たないうちに，インベーダーへの攻撃が不可能になってしまいます。

？？：フム，フム。

ツカ：ところで，本当に編三さんはどこに行ってしまったのでしょうね？

GAMEの完成

やりました！ ついに**GAMEが完成**しましたよ。次の**リスト4—14**で。いちおう

ALL BASIC版

でマア，マアのGAMEができました。スピードも，**あれだけのキャラクタ**を登場させたわりには，マアマアの速さが得られました。したがって次の**リスト4—14**までマスターしていただければ，いちおう

リアルタイムGAMEの基本は卒業した

ことになります。なぜなら，この

インベーダー・パニック

は，**リアルタイムGAMEの基本をすべて含む**ことができたからです。

それでは，さっそくその**リスト4—14**を走らせてみることにしましょう。なおGAMEのルールは，暫定ルールをほとんど変えないで済みました。

変更点： ジュニア・インベーダーをすべて消したときのボーナス点を少し変更した。

追加点： インベーダー，ジュニア・インベーダーがミサイルで攻撃してくる（もちろんこれがなければ，GAMEになりませんね)。

したがって**ゲーム・オーバー**は，

ビーム砲が3台ともやられたとき

となります。

写真13を御覧ください。インベーダー，ジュニア・インベーダーとの激しい戦闘の模様です。ビームとミサイルが飛びかっていますね。敵の**ミサイルは，同時に3発まで**存在できますが，**ビームは1発**だけです。**写真14**は，ビーム砲がやられたところです。ちゃんと

ビーム砲が爆発

したように見えますから，ぜひプログラムを入力して走らせてみてください。

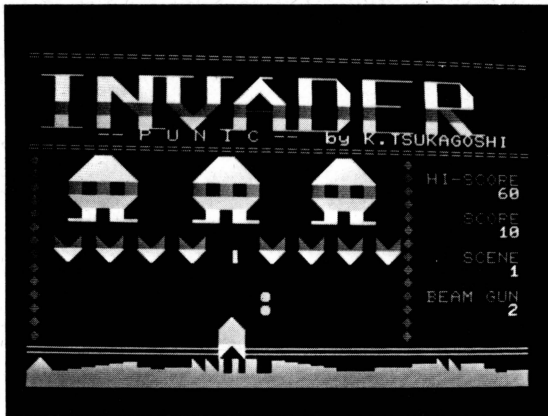
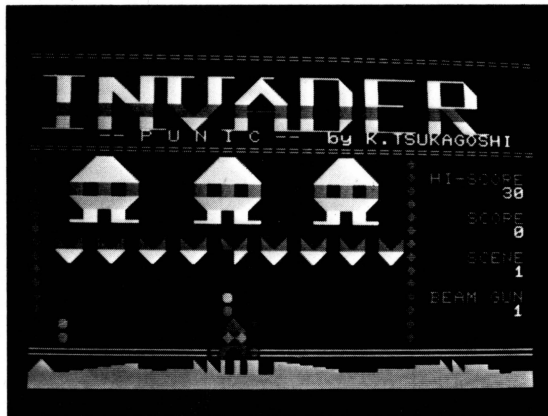
《リスト4-14》インベーダー・パニック完成(？)

```
1000 '============================
1010 '   INVADER PANIC --list 14--
1020 '        1982.5.4-?.??
1030 '               by K.TSUKAGOSHI
1040 '============================
1050 '
1060 '—— VARIABLE ——
1070 'IO,I9                :VALUE OF INP
1080 'BMAX                 :MAX OF YBEAM
1090 'XBEAM,YBEAM          :LOCATE OF BEAM (IF YBEAM=0 TEHN NO BEAM)
1100 'XGUN ,YGUN           :LOCATE OF BEAM GUN
1110 'XIV$                 :LOCATE X OF INVADER
1120 'XJIV$                :LOCATE X OF JUNIOUR INVADER
1130 'NMS                  :POINTER OF MISSILE NUMBER (USED IN 2400-2440)
1140 'LMS                  :LEFT OF MISSILE (MAX=3)
1150 'LIV                  :LEFT OF INVADER
1160 'LJIV                 :LEFT OF JUNIOUR INVADER
1170 'ABLE                 :IF ABLE=0 TEHN IT IS IMPOSSIBLE TO ATTACK INVADER.
1180 'HSCR                 :HI-SCORE
1190 'SCR                  :SCORE
1200 'SN                   :SCENE
1210 'LBGUN                :LEFT OF BEAM GUN
1220 'GAME                 :IF GAME=0 THEN GAME OVER.
1230 'CLK                  :CLOCK COUNTER
1240 '
1250 '—— DIMENSION ——
1260 'IV$(4)               :INVADE
1270 'JI$(1)               :JUNIOR INVADER
1280 'XMS(4),YMS(4)        :LOCATE OF MISSILE
1290 '
1300 '---------
1310 '   MAIN
1320 '---------
1330 '
1340 '== COLD START ==
```

```
1350 DEFINT A-Z'                                       INTEGER MODE
1360 WIDTH 40,25:CONSOLE 0,25,0,1'                     SET TV MODE
1370   BMAX=8:HSCR=0:CLK=0
1380   DIM I$(4),IV$(4),J$(1),XMS(4),YMS(4)
1390 '
1400 '== HOT START ==
1410 COLOR 7:PRINT CHR$(12);'                          CLEAR
1420   GAME=1:SCR=0:SN=1'                              RESET GAME STATUS
1430   XGUN=13:YGUN=20:YBEAM=0:LBGUN=3
1440   GOSUB 2030'                                     RESET INVADER DATA
1450   GOSUB 2130'                                     RESET JUNIOUR INVADER DATA
1460   GOSUB 1730'                                     CALL PRINT GAME ARER
1470   GOSUB 2350'                                     CALL PRINT BEAM GUN
1480 '
1490 '== GAME ==
1500 IF GAME=0 THEN 1650'                              game end ?
1510   CLK=CLK+1:IF CLK=3 THEN CLK=0'                  COUNTUP CLOCL
1520   GOSUB 2470'                                     KEYSCAN
1530   IF INT(RND(1)*50)<10 THEN GOSUB 3150'           1/5 CALL MOVE INVADER
1540   GOSUB 2840'                                     CALL MOVE BEAM
1550   IF BMAX<YBEAM AND YBEAM <13 THEN GOSUB 3450' CHECK IV
1560   IF YBEAM=15 THEN GOSUB 3300'                    CHECK JIV
1570   GOSUB 2470'                                     KEYSCAN
1580   IF CLK=0 THEN GOSUB 3230'                       CALL MOVE JUNIOUR INVADER
1590   GOSUB 2690'                                     HIT MISSILE
1600   GOSUB 2990'                                     MOVE MISSILE
1610   GOSUB 3610'                                     CHECK OF CRASH OF BEAM BUN
1620 GOTO 1500
1630 '
1640 '== GAME END ==
1650   IF SCR>HSCR THEN HSCR=SCR
1660 GOTO 1410
1670 '
1680 '-------
1690 '  SUB
1700 '-------
1710 '
1720 '== PRINT GAME AREA ==
1730 COLOR 1
1740 PRINT "=======================================";
1750 COLOR 7
1760 PRINT " ___    ____ __ __ ___  ____ ___       ";
1770 PRINT " ■ ■■■ ■ ▲ ■ ■■ ■  ■ ■         ";
1780 PRINT " ■ ■■■ ■ ■▲■ ■ ■■_  ■■■         ";
1790 COLOR 2
1800 PRINT " ■ ■■■■■ ■■■■ ■ ■■  ■■         ";
1810 COLOR 3
1820 PRINT " ■■ ■ ■_ ▼ ■ ■■■■ ■_ ■■_          ";
1830 COLOR 6:PRINT "       -- P U N I C --   ";
1840 COLOR 7:PRINT "by K.TSUKAGOSHI  "
1850 COLOR 1
1860 PRINT "=======================================";
1870 LOCATE 0,22
1880 COLOR 5:PRINT "=======================================";
1890 COLOR 6:PRINT "▲  ▄▄  ▄▄■■■ ■▄▄▄   ▄▄▄■▄▄   _";
1900 LINE (0,24)-(38,24),"■",6,BF
1910 COLOR 2:FOR Y=8 TO 21
1920           LOCATE 0,Y:PRINT "◆";:LOCATE 28,Y:PRINT "◆";
1930         NEXT
1940 COLOR 4:LOCATE 30, 9:PRINT "HI-SCORE";:GOSUB 3950
1950 COLOR 4:LOCATE 30,12:PRINT "   SCORE";:GOSUB 4000
1960 COLOR 4:LOCATE 30,15:PRINT "   SCENE";:GOSUB 4050
1970 COLOR 4:LOCATE 30,18:PRINT "BEAM GUN";:GOSUB 4100
1980 GOSUB 2350'                                       PRINT BEAM GUN
1990 GOSUB 2220'                                       PRINT INVADER
2000 GOTO  2300'                                       PRINT JUNIOUR INVADER
2010 '
2020 '== INITIAL INVADER DATA ==
2030 RESTORE 4180
2040 FOR I=0 TO 4
2050   READ I$(I):IV$(I)=""
2060   FOR J=1 TO 3'                                   IV$=I$*3
```

```
2070      IV$(I)=IV$(I)+I$(I)
2080 NEXT J,I
2090 XIV$ ="1121100001121100001121100000"'        RESET XIV$
2100 LIV=3:RETURN
2110 '
2120 '== INITIAL JUNIOUR INVADER DATA ==
2130 JI$(0)="":JI$(1)=""'                        RESET J$( )
2140 FOR I=1 TO 9
2150   JI$(0)=JI$(0)+"◣◢ "
2160   JI$(1)=JI$(1)+"▼ "
2170 NEXT
2180 XJIV$="110110110110110110110110110"'        RESET XJIV$
2190 LJIV=9:ABLE=0:RETURN
2200 '
2210 '== PRINT INVADER ==
2220   COLOR 7:LOCATE 1,8 :PRINT IV$(0);
2230   COLOR 7:LOCATE 1,9 :PRINT IV$(1);
2240   COLOR 2:LOCATE 1,10:PRINT IV$(2);
2250   COLOR 3:LOCATE 1,11:PRINT IV$(3);
2260   COLOR 7:LOCATE 1,12:PRINT IV$(4);
2270 RETURN
2280 '
2290 '== PRINT JUNIOR INVADER ==
2300   COLOR 2:LOCATE 1,14:PRINT JI$(0);
2310   COLOR 7:LOCATE 1,15:PRINT JI$(1);
2320 RETURN
2330 '
2340 '== PRINT BEAM GUN ==
2350   COLOR 1:LOCATE XGUN,YGUN   :PRINT " ▲ ";
2360           LOCATE XGUN,YGUN+1:PRINT " ■ ";
2370   COLOR 5:LOCATE XGUN,YGUN+2:PRINT "⊐◢◣⊏";
2380 RETURN
2390 '
2400 '== PRINT MISSILE ==
2410   COLOR 4
2420     LOCATE XMS(NMS),YMS(NMS)   :PRINT "●";
2430     LOCATE XMS(NMS),YMS(NMS)+1:PRINT "●";
2440 RETURN
2450 '
2460 '== KEY SCAN ==
2470   I0=INP(0):I9=INP(9)'                     KEYSCAN
2480   IF I0=255 AND I9=255 THEN RETURN'        NO TOUCH TEHN RET
2490   IF I0=239 THEN 2550'                     LEFT ?
2500   IF I0=191 THEN 2590'                     RIGHT ?
2510   IF I9=191 THEN 2630'                     SPACE ?
2520 RETURN'                                    OTHE KEY THEN RET
2530 '
2540 '== MOVE LEFT BEAM GUN ==
2550 IF XGUN<2 THEN RETURN'                     LEFT EDGE ?
2560 XGUN=XGUN-1:GOTO 2340
2570 '
2580 '== MOVE RIGHT BEAM GUN ==
2590 IF XGUN>23 THEN RETURN'                    RIGHT EDGE ?
2600 XGUN=XGUN+1:GOTO 2340
2610 '
2620 '== SHOOT BEAM GUN ==
2630 IF YBEAM<>0 THEN RETURN'                   EXIST BEAM
2640   XBEAM=XGUN+2:YBEAM=19'                   INITIAL (XBEAM,YBEAM)
2650   LOCATE XBEAM,YBEAM:PRINT "▌";'           SHOOT BEAM
2660   RETURN
2670 '
2680 '== SHOOT MISSILE ==
2690 IF LMS=3 THEN RETURN'                      MISSILE FUL !
2700   LMS=LMS+1'                               INCREMENT MISSILE
2710   IF LJIV=0 THEN 2780'                     NO JUNIOUR INVADER
2720     IF RND(1)*100>60 THEN XMS(LMS)=XGUN+1 ELSE XMS(LMS)=INT(RND(1)*27+1)
2730     IF MID$(XJIV$,XMS(LMS),1)="0" THEN 2790
2740       YMS(LMS)=16
2750       NMS=LMS:GOSUB 2410:RETURN'           PRINT MISSILE
2760 '
2770 == INVADER HITS MISSILE ==
2780 IF RND(1)*100>60 THEN XMS(LMS)=XGUN+1 ELSE XMS(LMS)=INT(RND(1)*27+1)
```

```
2790    IF MID$(XIV$,XMS(LMS),1)="0" THEN LMS=LMS-1:RETURN
2800      YMS(LMS)=13
2810      NMS=LMS:GOSUB 2410:RETURN'              PRINT MISSILE
2820 '
2830 '== MOVE BEAM ==
2840 IF YBEAM=0 THEN RETURN'                     NO BEAM
2850    COLOR 6
2860    LOCATE XBEAM,YBEAM:PRINT " ";'           ERASE BEAM
2870    YBEAM=YBEAM-2'                           BEAM UP
2880    IF YBEAM<BMAX THEN 2930'                 ERASE BEAM
2890      LOCATE XBEAM,YBEAM:PRINT "I";'         NEW BEAM
2900      RETURN
2910 '
2920 '== ERASE BEAM ==
2930 YBEAM=0:IF ABLE=0 THEN RETURN'              BEAM END
2940 ABLE=ABLE-1'                                COUNT DOWN ABLE
2950 IF ABLE=0 THEN GOSUB 2130'                  RESURRECDTIN JUNIOUR INBVADER
2960    RETURN
2970 '
2980 '== MOVE MISSILE ==
2990 IF LMS=0 THEN RETURN'                       NO MISSILE
3000    COLOR 4
3010    I=1
3020    IF I>LMS THEN RETURN
3030      LOCATE XMS(I),YMS(I)  :PRINT " ";'     ERASE MISSILE
3040      LOCATE XMS(I),YMS(I)+1:PRINT " ";
3050      YMS(I)=YMS(I)+2:IF YMS(I)<21 THEN 3100'DON'T REACH ?
3060        FOR J=I TO LMS'                      REACH
3070          XMS(J)=XMS(J+1):YMS(J)=YMS(J+1)'   MOVE DATA
3080        NEXT
3090        LMS=LMS-1:GOTO 3120
3100      LOCATE XMS(I),YMS(I)  :PRINT "●";'     PRINT NEW MISSILE
3110      LOCATE XMS(I),YMS(I)+1:PRINT "●";
3120    I=I+1:GOTO 3020
3130 '
3140 '== MOVE INVADER ==
3150 FOR I=0 TO 4
3160    IV$(I)=RIGHT$(IV$(I),2)+LEFT$(IV$(I),25)
3170 NEXT
3180 GOSUB 2220'                                 PRINT INVADER
3190 XIV$=RIGHT$(XIV$,2)+LEFT$(XIV$,25)'         LOCATE X
3200 RETURN
3210 '
3220 '== MOVE JUNIOUR INVADER ==
3230 JI$(0)=RIGHT$(JI$(0),26)+LEFT$(JI$(0),1)
3240 JI$(1)=RIGHT$(JI$(1),26)+LEFT$(JI$(1),1)
3250 GOSUB 2300'                                 PRINT JUNIOUR INVADER
3260 XJIV$=RIGHT$(XJIV$,26)+LEFT$(XJIV$,1)'      LOCATE XJUNIOURINVADER
3270 RETURN
3280 '
3290 '== CHECK CRUSH OF JUNIOUR INVADER ==
3300 IF ABLE>0 THEN RETURN
3310 IF MID$(XJIV$,XBEAM,1)="0" THEN RETURN'     MISS !
3320    I=XBEAM'                                 SEARCH FOR LEFT OF 1
3330      IF MID$(XJIV$,I-1,1)="1" THEN I=I-1
3340    MID$(JI$(0),I)="  "'                     DELETE JUNIOUR INVADER
3350    MID$(JI$(1),I)="  "
3360    MID$(XJIV$,I)="00"
3370    GOSUB 2930'                              ERASE BEAM
3380    SCR=SCR+10:GOSUB 4000'                   SCORE COUNT UP
3390    LJIV=LJIV-1:IF LJIV>0 THEN RETURN'
3400      ABLE=5'                                ATTACK INVADER OK !
3410      SCR=SCR+INT(RND(0)*21+10)*10:GOSUB 4000' 100-300 STEP 10
3420      RETURN
3430 '
3440 '== CHECK CRUSH OF INVADER ==
3450 IF ABLE=0 THEN RETURN'                      IMPOSSIBLE
3460 IF MID$(XIV$,XBEAM,1)<>"2" THEN RETURN'     MISS !
3470    I=XBEAM'                                 SEARCH FOR LEFT OF 1
3480      IF MID$(XIV$,I-1,1)="0" THEN 3490 ELSE I=I-1:GOTO 3480
3490    FOR J=0 TO 4
3500      MID$(IV$(J),I)="      "
```

```
3510    NEXT
3520    MID$(XIV$,I)="00000"
3530    SCR=SCR+100:GOSUB 4000'                    SCORE COUNT UP
3540    LIV=LIV-1:IF LIV>0 THEN RETURN'            EXIST INVADER
3550      SCR=SCR+1000:GOSUB 4000'                 SCORE COUNT UP
3560      SN=SN+1:GOSUB 4050'                      SCENE COUNT UP
3570      GOSUB 2030:GOSUB 2130'                   RESET IV,JIV DATA
3580      RETURN
3590 '
3600 '== CHECK CRUSH OF BEAM GUN ==
3610 I=1
3620 IF I>LMS THEN RETURN'
3630    IF YMS(I)<19 THEN 3750'                    DON'T REACH
3640    IF XMS(I)<XGUN+1 OR XMS(I)>XGUN+2 THEN 3750' SAFE
3650      GOSUB 3790'                              CRASH BEAM GUN
3660      LOCATE XMS(I),YMS(I)  :PRINT " ";
3670      LOCATE XMS(I),YMS(I)+1:PRINT " ";
3680      GOSUB 2350'                              PRINT BEAM GUN
3690      FOR J=I TO LMS'                          ERASE MISSILE
3700        XMS(J)=XMS(J+1):YMS(J)=YMS(J+1)
3710      NEXT
3720      LMS=LMS-1
3730      LBGUN=LBGUN-1:IF LBGUN=0 THEN 3760'      GAME END ?
3740      GOSUB 4100'                              PRINT LEFT OF BEAM GUN
3750 I=I+1:GOTO 3620
3760 GAME=0:RETURN'                                GAME END
3770 '
3780 '== CRUSH OF BEAM GUN ==
3790 COLOR 2
3800    LOCATE XGUN,YGUN  :PRINT "    ";
3810    LOCATE XGUN,YGUN+1:PRINT " ◆◆ ";
3820    LOCATE XGUN,YGUN+2:PRINT "    ";
3830      FOR J=0 TO 100:NEXT
3840    LOCATE XGUN,YGUN  :PRINT " ◆◆ ";
3850    LOCATE XGUN,YGUN+1:PRINT "    ";
3860    LOCATE XGUN,YGUN+2:PRINT " ◆◆ ";
3870      FOR J=0 TO 100:NEXT
3880    LOCATE XGUN,YGUN  :PRINT "◆  ◆";
3890    LOCATE XGUN,YGUN+1:PRINT " ◆◆ ";
3900    LOCATE XGUN,YGUN+2:PRINT "◆  ◆";
3910      FOR J=0 TO 100:NEXT
3920 RETURN
3930 '
3940 '== PRINT HI-SCORE ==
3950    COLOR 7
3960    LOCATE 33,10:PRINT USING "#####";HSCR;
3970 RETURN
3980 '
3990 '== PRINT SCORE ==
4000    COLOR 7
4010    LOCATE 33,13:PRINT USING "#####";SCR;
4020 RETURN
4030 '
4040 '== PRINT SCENE ==
4050    COLOR 7
4060    LOCATE 35,16:PRINT USING "###";SN;
4070 RETURN
4080 '
4090 '== PRINT LEFT OF BEAM GUN ==
4100    COLOR 7
4110    LOCATE 37,19:PRINT USING "#";LBGUN-1;
4120 RETURN
4130 '
4140 '-------
4150 '  DATA
4160 '-------
4170 '
4180 DATA " ◢■◣    ":'                               INVADER DATA
4190 DATA "◢■■■◣   "
4200 DATA "■ ■ ■   "
4210 DATA "◥■■■◤   "
4220 DATA "_■ ■_   "
```

《写真13》インベーダー軍団との戦い

《写真14》ビーム砲爆発！

変数のまとめ

それでは，その最後の**リスト4―14**を解析していきましょう。

ミサイルの処理

が追加されただけですから，**リスト4―13**まで御理解いただけた方なら，簡単にわかります。

ミサイル処理は，大きく

2680〜2750：ミサイルを発射する

2980〜3120：ミサイルを動かす

の二つに分かれます。そしてそれぞれメインルーチンの1590行，1600行でＣＡＬＬされているわけです。

これら二つのルーチンを理解するため，最初にミサイル関数の変数をまとめておきましょう。

① **ＬＭＳ**

理在発射されているミサイルの数を表わします。ミサイルは最大３発まで発射されますから，

ＬＭＳ＝３

ならそれ以上ミサイルを発射させません。また

ＬＭＳ＝０

なら，現在ミサイルは１発も存在しないことを示しています。

② **ＸＭＳ（Ｘ），ＹＭＳ（Ｘ）**

Ｘ番目のミサイルの現在置です。実際には，

ＸＭＳ（１），ＹＭＳ（１）：１番目

ＸＭＳ（２），ＹＭＳ（２）：２番目

ＸＭＳ（３），ＹＭＳ（３）：３番目

の三つが使われます。

③ **ＮＭＳ**

ミサイルをプリントするサブルーチンをＣＡＬＬするときのパラメータに使います。ミサイルは，今見たように

１，２，３

の３種類があります。そこで表示したいミサイルの番号をＮＭＳに入れて表示ルーチンをＣＡＬＬすると，そのミサイルが表示されます。

ミサイル発射ルーチン

変数の整理が終わりましたので，実際のプログラムリストを見て行きましょう。最初は，2680行からの

ミサイル発射ルーチン

です。ここの流れは，次のようになっています。

①　ＬＭＳ＝３

のときは，**ミサイルは飽和状態**ですから，それ以上は発射しません（2690行)。

② ミサイル発射ルーチンは，大きく

インベーダーから落とす

ジュニア・インベーダーから落とす

の**２種類**に分かれます。その区分けは，次の基準で行っています。

1) **ジュニア・インベーダーがいない**，すなわち

ＬＪＩＶ＝０

のときは，**かならずインベーダーの下**から落とす（2710行)。

2) それ以外のときは，まず**ジュニア・インベーダーに優先権**をもたせます。そして乱数で落とす位置を決めますが，もしそこにジュニア・インベーダーがいないときは，インベーダーの下から落とすことにします。インベーダーもそこにいないときは，結局**不発**です（2730行，2790行)。

③ ミサイルの位置を決める乱数の使い方は，

五分の二：ビーム砲の真上

五分の三：乱数で決める

の確率で発生させています（2720行）。

④　ミサイルを発射させたときは，ミサイルの数

LMSをカウント・アップ

し，ミサイルの位置を

ヨコ座標：XMS（LMS）
→③で決めた値
タテ座標：YMS（LMS）
→ インベーダーから＝13
ジュニア・インベーダーから＝16

のように初期化してやります。

ミサイルの移動処理

次は，ミサイルの移動ルーチンです。

ここは今までの各種移動ルーチンに比べ，少し複雑です。というのは，動かす

ミサイルの数が複数

であり，しかも始末の悪いことにその数が

0～3と不定

だからです。そのためミサイルの現在数を表わす

変数LMS

が大活躍することになります。

まず，

LMS＝0（ミサイル無し）

なら何もしません（2990行）。そして

1番目のミサイルから

LMS番目のミサイルまで

を調べるわけです。それが3020行と3120行で作っているループで，**1回ループするたびに各ミサイルが一つずつ動く**わけです。

ループの中身は，今までとほとんど同じですからとくに問題はないと思いますが，ミサイルが真下まで来て消滅するところの処理（3060～3090行）が少し難しいかもしれません。そこで**第4—20図**を使って説明してみましょう。

第4—20図①のように仮に3発のミサイルがあったとします。このとき，

LMS＝3

です。そしてその中の1番目に記憶されているミサイルが，今真下に届き消してやるとします。すると②のように

1番目のミサイルのDATAは不要

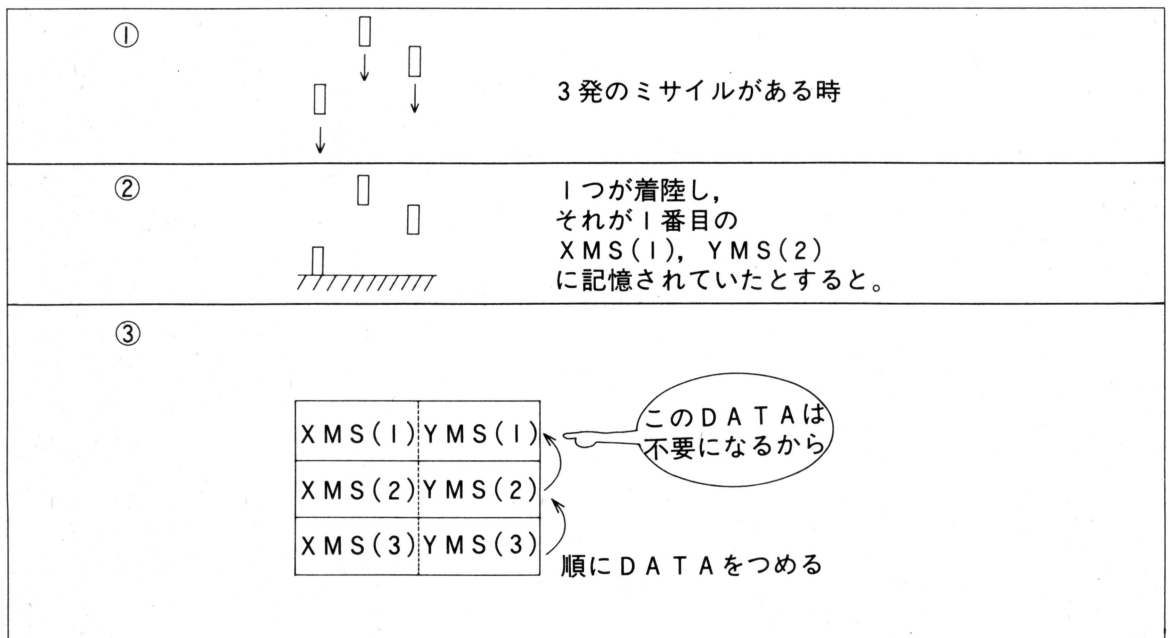
になります。そこで2番目～3番目のミサイルのデータを**順送りに前につめてやります**。その結果③のように

XMS＝（3），YMS（3）

のところは空きになり，

LMS＝2

となります。

3060行～3080行は，以上の操作をしてやっているわけです。これにより，**前のミサイルのDATAをプログラムの中から抹殺**することができます。良く図に書いて考えてみてください。

《第4—20図》ミサイルの移動処理

GAME終了処理

最後に**ビーム砲がインベーダーのミサイルにやられたかをチェックするルーチン**を調べてみましょう 3600行～3760行)。このルーチンは，メインルーチンの1610行からCALLされます。

① このルーチンも

　　0～3のミサイル

を個別にチェックします。3620行と3750行で作っているルーチンがそれです。

② チェックは，**ミサイルがビーム砲の位置に到達**すると，すなわち

　　YMS (ミサイル番号)≧18

になると開始されます (3630行)。

③ チェックは，ミサイルのヨコ座標とビーム砲のヨコ座標を比較して行います (3640行)。

④ 当たった場合の処理は，次の通りです。

1) ビーム砲を消す (3650～3680行)。
2) ミサイルを消す (3690～3710行)。
3) ビーム砲の数を減らす (3720行)。

⑤ ④の処理で

　　LBGUN＝0

すなわち**すべてのビーム砲がやられたときは，**

　　GAME OVER!

ですから，**GAME終了処理**を行います。

1) ゲーム進行フラグGAMEを，

　　GAME＝0

にします。(3760行)。このGAMEという変数は**GAME進行中は1**になっており，**GAME終了と同時に0**となります。

2) メインルーチンでは，1500行でこの変数の値を常にチェックしており，

　　GAME＝0

を確認すると，GAME終了ルーチン (1640行) へジャンプさせます。

3) ゲーム終了ルーチンでは，

　　得点とそれまでの最高点を比較

し，もし得点の方が大きければ，最高点を更新した上で

　　HOT START (1400行)

にジャンプさせます (1650行～1660行)。

おまけ

以上，**リスト4—14**で〝インベーダー・パニック〟は，

　　GAMEとして完成

したことになります。いかがでしたか？エッ？ **物足りない？**

ごもっともだと思います。**リスト4—14は，リアルタイムGAMEの教材**として作ったものです。したがって

　　GAMEの基礎

はすべて盛り込んでありますが，**それ以上は何も入っていません。**したがってGAMEとして見ると，物足りなさを感じるかもしれません。たとえば**リスト4—14**を走らせると，たしかにGAMEはできるのですが，

- ジュニア・インベーダーを消しても，ただ消えるだけ。
- ビーム砲がやられても，すぐ次のゲームが始まってしまう。
- GAME OVERの表示がない。

等の不満が残るでしょう。

もちろんこれらの不満を解消するのは，簡単です。足りない面をプログラム化して，**リスト4—14**に追加してやれば良かったのです。でもそうすると，どうなるでしょうか？ **リスト4—14は，益々長くなってしまいます。**長くなると，理解するのによけいな時間がかかってしまうでしょう。教材としてのプログラムは，それではいけないのです。やはり基礎的なものの

　　エキスだけ

にすべきです。

しかし，やはり不満なものは不満ですね？ そこで

　　おまけ

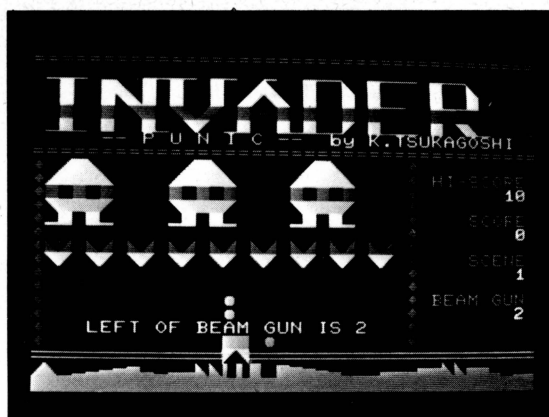
として**二つのプログラム**を用意しました。まず**リスト4—15**です。これは，

　　オールBASIC版

として**リスト4—14**を発展させたものです。ジュニア・インベーダーやインベーダーを消すと，ちゃんと爆発します。ビーム砲がやられても同じです。そしてビーム砲の残り数が表示されます (**写真15**)。ゲーム・オーバーももちろんきちんと表示されます (**写真16**)。

これでも何か物足りませんね？ なぜでしょう？

　　音？

そうです。今までのプログラムは，すべて音が入っていませんでした。GAMEに音を入れようとすると，各自のマシンによって様子がだいぶ変わってきます。音楽機能のついているもの，いないもの。まったく音の出ないもの等いろいろですね。したがって，音につ

《写真15》ビーム砲の残り数も表示

《写真16》GAME　OVER処理

《リスト4-15》爆発処理，GAME　OVER処理等追加

```
1000 '============================
1010 '  INVADER PANIC --list 15--
1020 '       1982.5.4-?.??
1030 '             by K.TSUKAGOSHI
1040 '============================
1050 '
1060 '--- VARIABLE ---
1070 'IO,I9               :VALUE OF INP
1080 'BMAX                :MAX OF YBEAM
1090 'XBEAM,YBEAM         :LOCATE OF BEAM (IF YBEAM=0 TEHN NO BEAM)
1100 'XGUN ,YGUN          :LOCATE OF BEAM GUN
1110 'XIV$                :LOCATE X OF INVADER
1120 'XJIV$               :LOCATE X OF JUNIOUR INVADER
1130 'NMS                 :POINTER OF MISSILE NUMBER (USED IN 2400-2440)
1140 'LMS                 :LEFT OF MISSILE (MAX=3)
1150 'LIV                 :LEFT OF INVADER
1160 'LJIV                :LEFT OF JUNIOUR INVADER
1170 'ABLE                :IF ABLE=0 TEHN IT IS IMPOSSIBLE TO ATTACK INVADER.
1180 'HSCR                :HI-SCORE
1190 'SCR                 :SCORE
1200 'SN                  :SCENE
1210 'LBGUN               :LEFT OF BEAM GUN
1220 'GAME                :IF GAME=0 THEN GAME OVER.
1230 'CLK                 :CLOCK COUNTER
1240 '
1250 '--- DIMENSION ---
1260 'IV$(4)              :INVADE
1270 'JI$(1)              :JUNIOR INVADER
1280 'XMS(4),YMS(4)       :LOCATE OF MISSILE
1290 '
1300 '--------
1310 '  MAIN
1320 '--------
1330 '
1340 '== COLD START ==
1350 DEFINT A-Z'                                  INTEGER MODE
1360 WIDTH 40,25:CONSOLE 0,25,0,1'                SET TV MODE
1370   BMAX=8:HSCR=0:CLK=0
1380   DIM I$(4),IV$(4),J$(1),XMS(4),YMS(4)
1390 '
1400 '== HOT START ==
1410 COLOR 7:PRINT CHR$(12);'                     CLEAR
1420   GAME=1:SCR=0:SN=1'                         RESET GAME STATUS
1430   XGUN=13:YGUN=20:YBEAM=0:LBGUN=3
1440   GOSUB 2090'                                RESET INVADER DATA
1450   GOSUB 2190'                                RESET JUNIOUR INVADER DATA
1460   GOSUB 1790'                                CALL PRINT GAME ARER
1470   GOSUB 2410'                                CALL PRINT BEAM GUN
1480 '
1490 '== GAME ==
1500 IF GAME=0 THEN 1650'                         game end ?
1510   CLK=CLK+1:IF CLK=3 THEN CLK=0'             COUNTUP CLOCL
```

```
1520    GOSUB 2530'                                   KEYSCAN
1530    IF INT(RND(1)*50)<10 THEN GOSUB 3210'         1/5 CALL MOVE INVADER
1540    GOSUB 2900'                                   CALL MOVE BEAM
1550    IF BMAX<YBEAM AND YBEAM <13 THEN GOSUB 3650' CHECK IV
1560    IF YBEAM=15 THEN GOSUB 3360'                  CHECK JIV
1570    GOSUB 2530'                                   KEYSCAN
1580    IF CLK=0 THEN GOSUB 3290'                     CALL MOVE JUNIOUR INVADER
1590    GOSUB 2750'                                   HIT MISSILE
1600    GOSUB 3050'                                   MOVE MISSILE
1610    GOSUB 4100'                                   CHECK OF CRASH OF BEAM BUN
1620 GOTO 1500
1630 '
1640 '== GAME END ==
1650 COLOR 6:LOCATE 4,15:PRINT "G A M E  O V E R ! !";
1660    FOR I=0 TO 2000:NEXT
1670 COLOR 5:LOCATE 6,17:PRINT "YOUR SCORE IS";SC
1680    FOR I=0 TO 2000:NEXT
1690 IF SCR<=HSCR THEN 1720'                          GET HI=SCORE ?
1700    HSCR=SCR
1710    COLOR 2:LOCATE 8,19:PRINT "IT'S HI-SCORE !"
1720 FOR I=0 TO 4000:NEXT:GOTO 1410
1730 '
1740 '-------
1750 '  SUB
1760 '-------
1770 '
1780 '== PRINT GAME AREA ==
1790 COLOR 1
1800 PRINT "=======================================";
1810 COLOR 7
1820 PRINT " ___  ____ __ _ ___ ____ ____      ";
1830 PRINT " ■ ■■■ ■ ▲■ ■■ ■ ■ ■▲        ";
1840 PRINT " ■ ■■■ ■ ▲■■ ■■_ ■_■        ";
1850 COLOR 2
1860 PRINT " ■ ■■ ■▼▼ ■■■ ■ ■  ■■        ";
1870 COLOR 3
1880 PRINT " ■ ■ ■_▼ ■ ■■▼■_ ■■        ";
1890 COLOR 6:PRINT "     -- P U N I C --  ";
1900 COLOR 7:PRINT "by K.TSUKAGOSHI "
1910 COLOR 1
1920 PRINT "=======================================";
1930 LOCATE 0,22
1940 COLOR 5:PRINT "════════════════════════════════";
1950 COLOR 6:PRINT "▲___▄▄__▲■■ ■▄▄___ __▄▲▄_ __";
1960 LINE (0,24)-(38,24),"■",6,BF
1970 COLOR 2:FOR Y=8 TO 21
1980          LOCATE 0,Y:PRINT "♦";:LOCATE 28,Y:PRINT "♦";
1990        NEXT
2000 COLOR 4:LOCATE 30, 9:PRINT "HI-SCORE";:GOSUB 4480
2010 COLOR 4:LOCATE 30,12:PRINT "   SCORE";:GOSUB 4530
2020 COLOR 4:LOCATE 30,15:PRINT "   SCENE";:GOSUB 4580
2030 COLOR 4:LOCATE 30,18:PRINT "BEAM GUN";:GOSUB 4630
2040 GOSUB 2410'                                      PRINT BEAM GUN
2050 GOSUB 2280'                                      PRINT INVADER
2060 GOTO  2360'                                      PRINT JUNIOUR INVADER
2070 '
2080 '== INITIAL INVADER DATA ==
2090 RESTORE 4710
2100 FOR I=0 TO 4
2110    READ I$(I):IV$(I)=""
2120    FOR J=1 TO 3'                                 IV$=I$*3
2130      IV$(I)=IV$(I)+I$(I)
2140 NEXT J,I
2150 XIV$ ="112110000112110000112110000"'             RESET XIV$
2160 LIV=3:RETURN
2170 '
2180 '== INITIAL JUNIOUR INVADER DATA ==
2190 JI$(0)="":JI$(1)=""'                             RESET J$( )
2200 FOR I=1 TO 9
2210    JI$(0)=JI$(0)+"▶◀ "
2220    JI$(1)=JI$(1)+"▼ "
2230 NEXT
```

```
2240 XJIV$="110110110110110110110110110"'        RESET XJIV$
2250 LJIV=9:ABLE=0:RETURN
2260 '
2270 '== PRINT INVADER ==
2280   COLOR 7:LOCATE 1,8 :PRINT IV$(0);
2290   COLOR 7:LOCATE 1,9 :PRINT IV$(1);
2300   COLOR 2:LOCATE 1,10:PRINT IV$(2);
2310   COLOR 3:LOCATE 1,11:PRINT IV$(3);
2320   COLOR 7:LOCATE 1,12:PRINT IV$(4);
2330 RETURN
2340 '
2350 '== PRINT JUNIOR INVADER ==
2360   COLOR 2:LOCATE 1,14:PRINT JI$(0);
2370   COLOR 7:LOCATE 1,15:PRINT JI$(1);
2380 RETURN
2390 '
2400 '== PRINT BEAM GUN ==
2410   COLOR 1:LOCATE XGUN,YGUN  :PRINT " ▲ ";
2420           LOCATE XGUN,YGUN+1:PRINT " ■ ";
2430   COLOR 5:LOCATE XGUN,YGUN+2:PRINT "=◢◣=";
2440 RETURN
2450 '
2460 '== PRINT MISSILE ==
2470   COLOR 4
2480     LOCATE XMS(NMS),YMS(NMS)  :PRINT "●";
2490     LOCATE XMS(NMS),YMS(NMS)+1:PRINT "●";
2500 RETURN
2510 '
2520 '== KEY SCAN ==
2530   I0=INP(0):I9=INP(9)'                       KEYSCAN
2540   IF I0=255 AND I9=255 THEN RETURN'          NO TOUCH TEHN RET
2550   IF I0=239 THEN 2610'                       LEFT ?
2560   IF I0=191 THEN 2650'                       RIGHT ?
2570   IF I9=191 THEN 2690'                       SPACE ?
2580 RETURN'                                      OTHE KEY THEN RET
2590 '
2600 '== MOVE LEFT BEAM GUN ==
2610 IF XGUN<2 THEN RETURN'                       LEFT EDGE ?
2620 XGUN=XGUN-1:GOTO 2400
2630 '
2640 '== MOVE RIGHT BEAM GUN ==
2650 IF XGUN>23 THEN RETURN'                      RIGHT EDGE ?
2660 XGUN=XGUN+1:GOTO 2400
2670 '
2680 '== SHOOT BEAM GUN ==
2690 IF YBEAM<>0 THEN RETURN'                     EXIST BEAM
2700   XBEAM=XGUN+2:YBEAM=19'                     INITIAL (XBEAM,YBEAM)
2710   LOCATE XBEAM,YBEAM:PRINT "❙";'              SHOOT BEAM
2720   RETURN
2730 '
2740 '== SHOOT MISSILE ==
2750 IF LMS=3 THEN RETURN'                        MISSILE FUL !
2760   LMS=LMS+1'                                 INCREMENT MISSILE
2770   IF LJIV=0 THEN 2840'                       NO JUNIOUR INVADER
2780     IF RND(1)*100>60 THEN XMS(LMS)=XGUN+1 ELSE XMS(LMS)=INT(RND(1)*27+1)
2790     IF MID$(XJIV$,XMS(LMS),1)="0" THEN 2850
2800       YMS(LMS)=16
2810       NMS=LMS:GOSUB 2470:RETURN'             PRINT MISSILE
2820 '
2830 == INVADER HITS MISSILE ==
2840 IF RND(1)*100>60 THEN XMS(LMS)=XGUN+1 ELSE XMS(LMS)=INT(RND(1)*27+1)
2850   IF MID$(XIV$,XMS(LMS),1)="0" THEN LMS=LMS-1:RETURN
2860     YMS(LMS)=13
2870     NMS=LMS:GOSUB 2470:RETURN'               PRINT MISSILE
2880 '
2890 '== MOVE BEAM ==
2900 IF YBEAM=0 THEN RETURN'                      NO BEAM
2910   COLOR 6
2920   LOCATE XBEAM,YBEAM:PRINT " ";'             ERASE BEAM
2930   YBEAM=YBEAM-2'                             BEAM UP
2940   IF YBEAM<BMAX THEN 2990'                   ERASE BEAM
2950     LOCATE XBEAM,YBEAM:PRINT "❙";'           NEW BEAM
```

```
2960      RETURN
2970 '
2980 '== ERASE BEAM ==
2990 YBEAM=0:IF ABLE=0 THEN RETURN'               BEAM END
3000 ABLE=ABLE-1'                                 COUNT DOWN ABLE
3010 IF ABLE=0 THEN GOSUB 2190'                   RESURRECDTIN JUNIOUR INBVADER
3020   RETURN
3030 '
3040 '== MOVE MISSILE ==
3050 IF LMS=0 THEN RETURN'                        NO MISSILE
3060   COLOR 4
3070   I=1
3080   IF I>LMS THEN RETURN
3090     LOCATE XMS(I),YMS(I)  :PRINT " ";'       ERASE MISSILE
3100     LOCATE XMS(I),YMS(I)+1:PRINT " ";
3110     YMS(I)=YMS(I)+2:IF YMS(I)<21 THEN 3160'DON'T REACH ?
3120       FOR J=I TO LMS'                        REACH
3130         XMS(J)=XMS(J+1):YMS(J)=YMS(J+1)'     MOVE DATA
3140       NEXT
3150       LMS=LMS-1:GOTO 3180
3160     LOCATE XMS(I),YMS(I)  :PRINT "●";'       PRINT NEW MISSILE
3170     LOCATE XMS(I),YMS(I)+1:PRINT "●";
3180   I=I+1:GOTO 3080
3190 '
3200 '== MOVE INVADER ==
3210 FOR I=0 TO 4
3220   IV$(I)=RIGHT$(IV$(I),2)+LEFT$(IV$(I),25)
3230 NEXT
3240 GOSUB 2280'                                  PRINT INVADER
3250 XIV$=RIGHT$(XIV$,2)+LEFT$(XIV$,25)'          LOCATE X
3260 RETURN
3270 '
3280 '== MOVE JUNIOUR INVADER ==
3290 JI$(0)=RIGHT$(JI$(0),26)+LEFT$(JI$(0),1)
3300 JI$(1)=RIGHT$(JI$(1),26)+LEFT$(JI$(1),1)
3310 GOSUB 2360'                                  PRINT JUNIOUR INVADER
3320 XJIV$=RIGHT$(XJIV$,26)+LEFT$(XJIV$,1)'       LOCATE XJUNIOURINVADER
3330 RETURN
3340 '
3350 '== CHECK CRUSH OF JUNIOUR INVADER ==
3360 IF ABLE>0 THEN RETURN
3370 IF MID$(XJIV$,XBEAM,1)="0" THEN RETURN'      MISS !
3380   I=XBEAM'                                   SEARCH FOR LEFT OF 1
3390     IF MID$(XJIV$,I-1,1)="1" THEN I=I-1
3400   MID$(JI$(0),I)="  "'                       DELETE JUNIOUR INVADER
3410   MID$(JI$(1),I)="  "
3420   MID$(XJIV$,I)="00"
3430   GOSUB 3520'                                CRASH JUNIOUR INVADER
3440   GOSUB 2990'                                ERASE BEAM
3450   SCR=SCR+10:GOSUB 4530'                     SCORE COUNT UP
3460   LJIV=LJIV-1:IF LJIV>0 THEN RETURN'
3470     ABLE=5'                                  ATTACK INVADER OK !
3480     SCR=SCR+INT(RND(0)*21+10)*10:GOSUB 4530' 100-300 STEP 10
3490     RETURN
3500 '
3510 '== CRUSH OF JUNIOUR INVADER ==
3520 COLOR 5
3530   FOR K=0 TO 1'                              SPARK
3540     LOCATE I,14:PRINT "◢ ";
3550     LOCATE I,15:PRINT " ◤";
3560       FOR J=0 TO 100:NEXT
3570     LOCATE I,14:PRINT " ◣";
3580     LOCATE I,15:PRINT "◥ ";
3590       FOR J=0 TO 100:NEXT
3600   NEXT
3610   LOCATE I,14:PRINT "  ";'                   ERASE
3620   LOCATE I,15:PRINT "  ";
3630 '
3640 '== CHECK CRUSH OF INVADER ==
3650 IF ABLE=0 THEN RETURN'                       IMPOSSIBLE
3660 IF MID$(XIV$,XBEAM,1)<>"2" THEN RETURN'      MISS !
3670   I=XBEAM'                                   SEARCH FOR LEFT OF 1
```

```
3680      IF MID$(XIV$,I-1,1)="0" THEN 3690 ELSE I=I-1:GOTO 3680
3690    FOR J=0 TO 4
3700      MID$(IV$(J),I)="         "
3710    NEXT
3720    MID$(XIV$,I)="00000"
3730    GOSUB 3830'                                     CRASH !
3740    GOSUB 2990'                                     ERASE BEAM
3750    SCR=SCR+100:GOSUB 4530'                         SCORE COUNT UP
3760    LIV=LIV-1:IF LIV>0 THEN RETURN'                 EXIST INVADER
3770      SCR=SCR+1000:GOSUB 4530'                      SCORE COUNT UP
3780      SN=SN+1:GOSUB 4580'                           SCENE COUNT UP
3790      GOSUB 2090:GOSUB 2190'                        RESET IV,JIV DATA
3800      RETURN
3810 '
3820 '== CRUSH OF INVADER ==
3830 COLOR 5
3840    LOCATE I, 8:PRINT "     ";
3850    LOCATE I, 9:PRINT " ◢◣ ";
3860    LOCATE I,10:PRINT "  ■  ";
3870    LOCATE I,11:PRINT " ◥◤ ";
3880    LOCATE I,12:PRINT "     ";
3890      FOR J=0 TO 120:NEXT
3900    LOCATE I, 8:PRINT "  ●  ";
3910    LOCATE I, 9:PRINT "  ●  ";
3920    LOCATE I,10:PRINT "●● ●●";
3930    LOCATE I,11:PRINT "  ●  ";
3940    LOCATE I,12:PRINT "  ●  ";
3950      FOR J=0 TO 120:NEXT
3960    LOCATE I, 8:PRINT "▲ ▲";
3970    LOCATE I, 9:PRINT "▼ ▼";
3980    LOCATE I,10:PRINT "  ◆  ";
3990    LOCATE I,11:PRINT "▲ ▲";
4000    LOCATE I,12:PRINT "▼ ▼";
4010      FOR J=0 TO 120:NEXT
4020    LOCATE I, 8:PRINT "     ";
4030    LOCATE I, 9:PRINT "     ";
4040    LOCATE I,10:PRINT "     ";
4050    LOCATE I,11:PRINT "     ";
4060    LOCATE I,12:PRINT "     ";
4070 RETURN
4080 '
4090 '== CHECK CRUSH OF BEAM GUN ==
4100 I=1
4110 IF I>LMS THEN RETURN'
4120    IF YMS(I)<19 THEN 4280'                         DON'T REACH
4130    IF XMS(I)<XGUN+1 OR XMS(I)>XGUN+2 THEN 4280' SAFE
4140      GOSUB 4320'                                   CRASH BEAM GUN
4150      LOCATE XMS(I),YMS(I)   :PRINT " ";
4160      LOCATE XMS(I),YMS(I)+1:PRINT " ";
4170      GOSUB 2410'                                   PRINT BEAM GUN
4180      FOR J=I TO LMS'                               ERASE MISSILE
4190        XMS(J)=XMS(J+1):YMS(J)=YMS(J+1)
4200      NEXT
4210      LMS=LMS-1
4220      LBGUN=LBGUN-1:IF LBGUN=0 THEN 4290'     GAME END ?
4230      COLOR 7:LOCATE 4,20:PRINT "LEFT OF BEAM GUN IS";LBGUN;
4240        FOR I=0 TO 4000:NEXT
4250      LOCATE 4,20:PRINT "                         "
4260      GOSUB 2410'                                   PRINT BEAM GUN
4270      GOSUB 4630'                                   PRINT LEFT OF BEAM GUN
4280 I=I+1:GOTO 4110
4290 GAME=0:RETURN'                                     GAME END
4300 '
4310 '== CRUSH OF BEAM GUN ==
4320 COLOR 2
4330    LOCATE XGUN,YGUN   :PRINT "    ";
4340    LOCATE XGUN,YGUN+1:PRINT " ◆◆ ";
4350    LOCATE XGUN,YGUN+2:PRINT "    ";
4360      FOR J=0 TO 100:NEXT
4370    LOCATE XGUN,YGUN   :PRINT " ◆◆ ";
4380    LOCATE XGUN,YGUN+1:PRINT "    ";
4390    LOCATE XGUN,YGUN+2:PRINT " ◆◆ ";
```

```
4400     FOR J=0 TO 100:NEXT
4410   LOCATE XGUN,YGUN  :PRINT "◆  ◆";
4420   LOCATE XGUN,YGUN+1:PRINT " ◆◆ ";
4430   LOCATE XGUN,YGUN+2:PRINT "◆  ◆";
4440     FOR J=0 TO 100:NEXT
4450 RETURN
4460 '
4470 '== PRINT HI-SCORE ==
4480   'COLOR 7
4490   LOCATE 33,10:PRINT USING "#####";HSCR;
4500 RETURN
4510 '
4520 '== PRINT SCORE ==
4530   COLOR 7
4540   LOCATE 33,13:PRINT USING "#####";SCR;
4550 RETURN
4560 '
4570 '== PRINT SCENE ==
4580   COLOR 7
4590   LOCATE 35,16:PRINT USING "###";SN;
4600 RETURN
4610 '
4620 '== PRINT LEFT OF BEAM GUN ==
4630   COLOR 7
4640   LOCATE 37,19:PRINT USING "#";LBGUN-1;
4650 RETURN
4660 '
4670 '--------
4680 '  DATA
4690 '--------
4700 '
4710 DATA "  ▲▲    ":'                        INVADER DATA
4720 DATA "▲▲▲▲▲   "
4730 DATA "■ ■ ■   "
4740 DATA "▼▼▼▼▼   "
4750 DATA "_■ ■_   "
```

いてはわざと避けてきました。

ところでＰＣ—8001もＢＥＥＰ音しか出ません。ピー，だけでは面白くありませんね。そのため次の**リスト４—16**では，**マシン語を導入**して音を出すようにしてあります（マシン語は，ＢＡＳＩＣが書き込むようになっています)。したがって**リスト４—16は，ＰＣ—8001の人しか使えません**。なにしろおまけですから，御勘弁ください。なお**リスト４—16**では，

ＧＡＭＥの説明

を追加しておきました（**写真17**)。この部分だけなら，他のマシンの人でも参考になると思います。まあ，これで一応ＧＡＭＥらしいＧＡＭＥが完成しました。

めでたし，めでたし。

《写真17》ＧＡＭＥ説明

〈糾弾コーナー〉

？？：——。

ツカ：長いことかかって

インベーダー・パニック

を作ったり，リストを解析したりしてきたわけですが，全体を通しまして何か御質問がありますでしょうか？

（一同）：シーン。

ツカ：それでは，よかったら一人ずつ感想をお聞かせください。

？？：鋭い質問ができなかったのは，残念です。

編長：〝ＧＡＭＩＮＧへの招待〞は，月刊「マイコン誌」の方でも連載しています。

＞君：あっ，そちらの方でも質問を受けつけていますのでドシドシどうぞ。なにせ「マイコン誌」は，**ア**

フター・サービスが売り物ですから。

編三：新しい枕を買ったから，早くお正月が来ないかなあ。来年こそ，もっといい初夢を見るぞ！

M子ちゃん：ＢＡＳＩＣって誰が作ったのかしら？今度〝ＧＡＭＩＧへの招待〟やる時，また私もいれてね？

第4章のおわりに

以上をもちまして，**インベーダー・パニック**の説明はすべて完了しました。もともとこのＧＡＭＥは，**リアルタイムＧＡＭＥ**をやさしく理解していただこうと思って製作したものです。したがってこのプログラムを解析していただければ，さらに複雑なＧＡＭＥへも挑戦することができるでしょう。このささやかな小文が，あなたのマイコン・ライフをより楽しいものにする一助となれば幸いです。

なおこのＧＡＭＥは，オールＢＡＳＩＣながらかなり大きなキャラクタを沢山登場させたり，またＧＡＭＥを面白くするためやや複雑なものとしたため，スピードが気になったかもしれません。もしこれ以上とスピードを望むのであれば，**マシン語**に頼らざるを得ないかもしれません。

《リスト4-16》インベーダー・パニック最終版

```
1000 '===========================
1010 '  INVADER PANIC --list 16--
1020 '       1982.5.4-6.6
1030 '             by K.TSUKAGOSHI
1040 '===========================
1050 '
1060 '-- VARIABLE --
1070 'IO,I9              :VALUE OF INP
1080 'BMAX               :MAX OF YBEAM
1090 'XBEAM,YBEAM        :LOCATE OF BEAM (IF YBEAM=0 TEHN NO BEAM)
1100 'XGUN ,YGUN         :LOCATE OF BEAM GUN
1110 'XIV$               :LOCATE X OF INVADER
1120 'XJIV$              :LOCATE X OF JUNIOUR INVADER
1130 'NMS                :POINTER OF MISSILE NUMBER (USED IN 2400-2440)
1140 'LMS                :LEFT OF MISSILE (MAX=3)
1150 'LIV                :LEFT OF INVADER
1160 'LJIV               :LEFT OF JUNIOUR INVADER
1170 'ABLE               :IF ABLE=0 TEHN IT IS IMPOSSIBLE TO ATTACK INVADER.
1180 'HSCR               :HI-SCORE
1190 'SCR                :SCORE
1200 'SN                 :SCENE
1210 'LBGUN              :LEFT OF BEAM GUN
1220 'CLK                :CLOCK COUNTER
1230 'GAME               :IF GAME=0 THEN GAME OVER.
1240 'STP                :IF STP=1 THEN GAME STOP.
1250 '
1260 '-- DIMENSION --
1270 'IV$(4)             :INVADE
1280 'JI$(1)             :JUNIOR INVADER
1290 'XMS(4),YMS(4)      :LOCATE OF MISSILE
1300 '
1310 '---------
1320 '  MAIN
1330 '---------
1340 '
1350 '== COLD START ==
1360 CLEAR 300,&HBFFF
1370   DEF USR1=&HC000'                      SOUND OF HIT BEAM
1380   DEF USR2=&HC006'                      SOUND OF CRASH JUNIOUR INVADER
1390   DEF USR3=&HC00C'                      SOUND OF CRASH INVADER
1400   DEF USR4=&HC012'                      SOUND OF CRASH BEAM GUN
1410     OUT &H51,0
1420     RESTORE 5330
1430     FOR I=&HC000 TO &HC078'             SET MACHINE LANGUAGE
1440       READ J$:POKE I,VAL("&H"+J$)
1450     NEXT
1460 DEFINT A-Z'                             INTEGER MODE
1470 WIDTH 40,25:CONSOLE 0,25,0,1'           SET TV MODE
1480   BMAX=8:HSCR=0:CLK=0:STP=0
1490   DIM I$(4),IV$(4),J$(1),XMS(4),YMS(4)
1500 '
1510 '== HOT START ==
1520 GOSUB 4880:IF STP=1 THEN END'           INSTRUCTION & GAME STOP
```

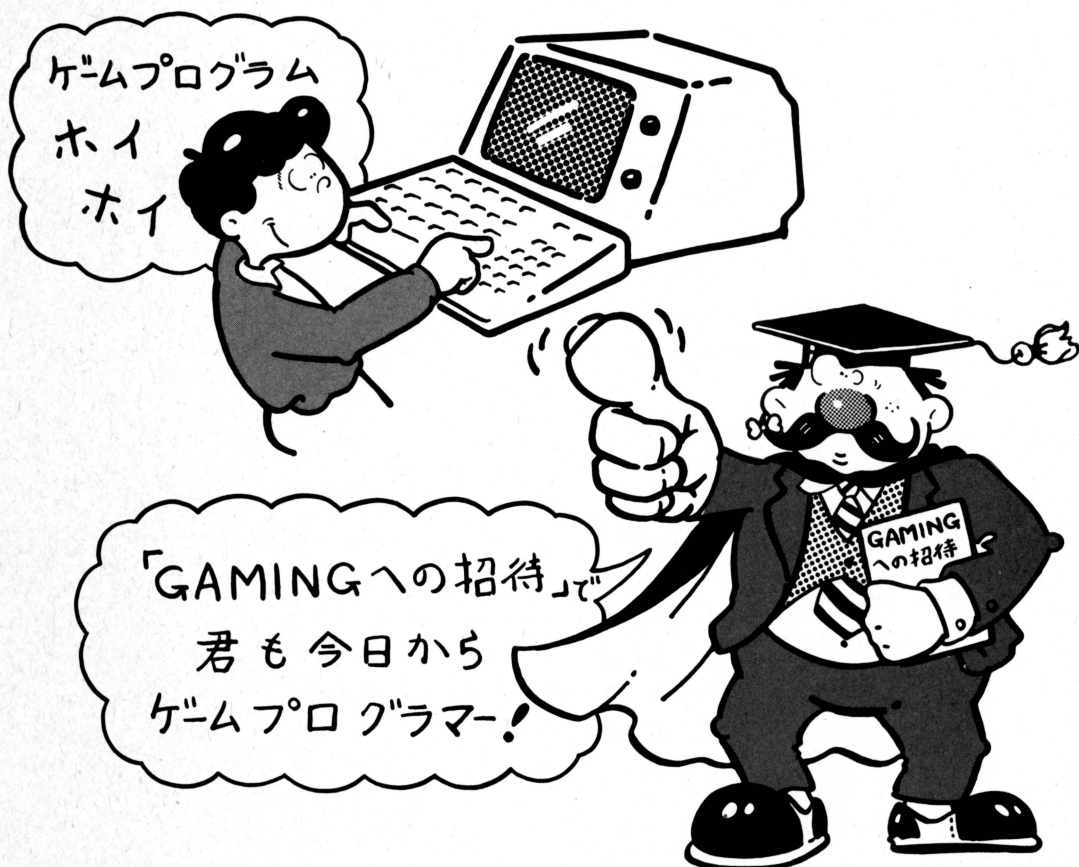
```
1530   COLOR 7:PRINT CHR$(12);'                        CLEAR
1540     GAME=1:SCR=0:SN=1'                            RESET GAME STATUS
1550     XGUN=13:YGUN=20:YBEAM=0:LBGUN=3
1560     GOSUB 2250'                                   RESET INVADER DATA
1570     GOSUB 2350'                                   RESET JUNIOUR INVADER DATA
1580     GOSUB 1910'                                   CALL PRINT GAME ARER
1590     GOSUB 2570'                                   CALL PRINT BEAM GUN
1600 '
1610 '== GAME ==
1620 IF GAME=0 THEN 1770'                              game end ?
1630   CLK=CLK+1:IF CLK=3 THEN CLK=0'                  COUNTUP CLOCL
1640   GOSUB 2690'                                     KEYSCAN
1650   IF INT(RND(1)*50)<10 THEN GOSUB 3380'           1/5 CALL MOVE INVADER
1660   GOSUB 3070'                                     CALL MOVE BEAM
1670   IF BMAX<YBEAM AND YBEAM <13 THEN GOSUB 3830' CHECK IV
1680   IF YBEAM=15 THEN GOSUB 3530'                    CHECK JIV
1690   GOSUB 2690'                                     KEYSCAN
1700   IF CLK=0 THEN GOSUB 3460'                       CALL MOVE JUNIOUR INVADER
1710   GOSUB 2920'                                     HIT MISSILE
1720   GOSUB 3220'                                     MOVE MISSILE
1730   GOSUB 4290'                                     CHECK OF CRASH OF BEAM BUN
1740 GOTO 1620
1750 '
1760 '== GAME END ==
1770 COLOR 6:LOCATE 4,15:PRINT "G A M E   O V E R ! !";
1780   FOR I=0 TO 2000:NEXT
1790 COLOR 5:LOCATE 6,17:PRINT "YOUR SCORE IS";SC
1800   FOR I=0 TO 2000:NEXT
1810 IF SCR<=HSCR THEN 1840'                           GET HI=SCORE ?
1820   HSCR=SCR
1830   COLOR 2:LOCATE 8,19:PRINT "IT'S HI-SCORE !"
1840 FOR I=0 TO 4000:NEXT:GOTO 1520
1850 '
1860 '-------
1870 '  SUB
1880 '-------
1890 '
1900 '== PRINT GAME AREA ==
1910 GOSUB 2080'                                       PRINT TITLE
1920 LOCATE 0,22
1930 COLOR 5:PRINT "══════════════════════════════════════";
1940 COLOR 6:PRINT "▲ ▁▁▃▅▃▁ ▁▁◢◢▌█ █ ██▆▃▁ ▁▁▃▅◢◣▃▃▁ ▁▁";
1950 LINE (0,24)-(38,24),"■",6,BF
1960 COLOR 2:FOR Y=8 TO 21
1970           LOCATE 0,Y:PRINT "◆";:LOCATE 28,Y:PRINT "◆";
1980         NEXT
1990 COLOR 4:LOCATE 30, 9:PRINT "HI-SCORE";:GOSUB 4680
2000 COLOR 4:LOCATE 30,12:PRINT "   SCORE";:GOSUB 4730
2010 COLOR 4:LOCATE 30,15:PRINT "   SCENE";:GOSUB 4780
2020 COLOR 4:LOCATE 30,18:PRINT "BEAM GUN";:GOSUB 4830
2030 GOSUB 2570'                                       PRINT BEAM GUN
2040 GOSUB 2440'                                       PRINT INVADER
2050 GOTO  2520'                                       PRINT JUNIOUR INVADER
2060 '
2070 '== PRINT TITLE ==
2080 PRINT CHR$(12);:COLOR 1
2090 PRINT "======================================";
2100 COLOR 7
2110 PRINT " ▁▁  ▁▁▁ ▁ ▁ ▁▁ ▁▁▁ ▁▁   ";
2120 PRINT "  ■ █◣██ █ ▲ █ ◣█  █ ◣   ";
2130 PRINT "  ■ █◥◣█ █◢◣ █ █▁▁ █▁█   ";
2140 COLOR 2
2150 PRINT "  ■ █◥█ ◥◢█▬█ █ █  █◣   ";
2160 COLOR 3
2170 PRINT " ▁█ █ █▁ ▼ ▁█ █▁█▁◤█▁▁ █◥▁  ";
2180 COLOR 6:PRINT "      -- P U N I C --  ";
2190 COLOR 7:PRINT "by K.TSUKAGOSHI "
2200 COLOR 1
2210 PRINT "======================================";
2220 RETURN
2230 '
2240 '== INITIAL INVADER DATA ==
```

```
2250 RESTORE 5270
2260 FOR I=0 TO 4
2270   READ I$(I):IV$(I)=""
2280   FOR J=1 TO 3'                                 IV$=I$*3
2290     IV$(I)=IV$(I)+I$(I)
2300 NEXT J,I
2310 XIV$ ="112110000112110000112110000"'          RESET XIV$
2320 LIV=3:RETURN
2330 '
2340 '== INITIAL JUNIOUR INVADER DATA ==
2350 JI$(0)="":JI$(1)=""'                          RESET J$( )
2360 FOR I=1 TO 9
2370   JI$(0)=JI$(0)+"▙▟ "
2380   JI$(1)=JI$(1)+"▼ "
2390 NEXT
2400 XJIV$="110110110110110110110110110"'          RESET XJIV$
2410 LJIV=9:ABLE=0:RETURN
2420 '
2430 '== PRINT INVADER ==
2440   COLOR 7:LOCATE 1,8 :PRINT IV$(0);
2450           LOCATE 1,9 :PRINT IV$(1);
2460   COLOR 2:LOCATE 1,10:PRINT IV$(2);
2470   COLOR 3:LOCATE 1,11:PRINT IV$(3);
2480   COLOR 7:LOCATE 1,12:PRINT IV$(4);
2490 RETURN
2500 '
2510 '== PRINT JUNIOR INVADER ==
2520   COLOR 2:LOCATE 1,14:PRINT JI$(0);
2530   COLOR 7:LOCATE 1,15:PRINT JI$(1);
2540 RETURN
2550 '
2560 '== PRINT BEAM GUN ==
2570   COLOR 1:LOCATE XGUN,YGUN  :PRINT " ▲ ";
2580           LOCATE XGUN,YGUN+1:PRINT " ■ ";
2590   COLOR 5:LOCATE XGUN,YGUN+2:PRINT "=▼▼=";
2600 RETURN
2610 '
2620 '== PRINT MISSILE ==
2630   COLOR 4
2640     LOCATE XMS(NMS),YMS(NMS)  :PRINT "●";
2650     LOCATE XMS(NMS),YMS(NMS)+1:PRINT "●";
2660 RETURN
2670 '
2680 '== KEY SCAN ==
2690   IO=INP(0):I9=INP(9)'                          KEYSCAN
2700   IF IO=255 AND I9=255 THEN RETURN'             NO TOUCH TEHN RET
2710   IF IO=239 THEN 2770'                          LEFT ?
2720   IF IO=191 THEN 2810'                          RIGHT ?
2730   IF I9=191 THEN 2850'                          SPACE ?
2740 RETURN'                                         OTHE KEY THEN RET
2750 '
2760 '== MOVE LEFT BEAM GUN ==
2770 IF XGUN<2 THEN RETURN'                          LEFT EDGE ?
2780 XGUN=XGUN-1:GOTO 2560
2790 '
2800 '== MOVE RIGHT BEAM GUN ==
2810 IF XGUN>23 THEN RETURN'                         RIGHT EDGE ?
2820 XGUN=XGUN+1:GOTO 2560
2830 '
2840 '== SHOOT BEAM GUN ==
2850 IF YBEAM<>0 THEN RETURN'                        EXIST BEAM
2860   XBEAM=XGUN+2:YBEAM=19'                        INITIAL (XBEAM,YBEAM)
2870   LOCATE XBEAM,YBEAM:PRINT "I ";'               SHOOT BEAM
2880   U=USR1(0)'
2890   RETURN
2900 '
2910 '== SHOOT MISSILE ==
2920 IF LMS=3 THEN RETURN'                           MISSILE FUL !
2930   LMS=LMS+1'                                    INCREMENT MISSILE
2940   IF LJIV=0 THEN 3010'                          NO JUNIOUR INVADER
2950     IF RND(1)*100>60 THEN XMS(LMS)=XGUN+1 ELSE XMS(LMS)=INT(RND(1)*27+1)
2960     IF MID$(XJIV$,XMS(LMS),1)="0" THEN 3020
2970       YMS(LMS)=16
```

```
2980       NMS=LMS:GOSUB 2630:RETURN'            PRINT MISSILE
2990 '
3000 == INVADER HITS MISSILE ==
3010 IF RND(1)*100>60 THEN XMS(LMS)=XGUN+1 ELSE XMS(LMS)=INT(RND(1)*27+1)
3020   IF MID$(XIV$,XMS(LMS),1)="0" THEN LMS=LMS-1:RETURN
3030     YMS(LMS)=13
3040     NMS=LMS:GOSUB 2630:RETURN'              PRINT MISSILE
3050 '
3060 '== MOVE BEAM ==
3070 IF YBEAM=0 THEN RETURN'                     NO BEAM
3080   COLOR 6
3090   LOCATE XBEAM,YBEAM:PRINT " ";'            ERASE BEAM
3100   YBEAM=YBEAM-2'                            BEAM UP
3110   IF YBEAM<BMAX THEN 3160'                  ERASE BEAM
3120     LOCATE XBEAM,YBEAM:PRINT "| ";'         NEW BEAM
3130     RETURN
3140 '
3150 '== ERASE BEAM ==
3160 YBEAM=0:IF ABLE=0 THEN RETURN'              BEAM END
3170 ABLE=ABLE-1'                                COUNT DOWN ABLE
3180 IF ABLE=0 THEN GOSUB 2350'                  RESURRECDTIN JUNIOUR INBVADER
3190   RETURN
3200 '
3210 '== MOVE MISSILE ==
3220 IF LMS=0 THEN RETURN'                       NO MISSILE
3230   COLOR 4
3240   I=1
3250   IF I>LMS THEN RETURN
3260     LOCATE XMS(I),YMS(I)  :PRINT " ";'      ERASE MISSILE
3270     LOCATE XMS(I),YMS(I)+1:PRINT " ";
3280     YMS(I)=YMS(I)+2:IF YMS(I)<21 THEN 3330'DON'T REACH ?
3290       FOR J=I TO LMS'                       REACH
3300         XMS(J)=XMS(J+1):YMS(J)=YMS(J+1)'    MOVE DATA
3310       NEXT
3320       LMS=LMS-1:GOTO 3350
3330     LOCATE XMS(I),YMS(I)  :PRINT "●";'      PRINT NEW MISSILE
3340     LOCATE XMS(I),YMS(I)+1:PRINT "●";
3350   I=I+1:GOTO 3250
3360 '
3370 '== MOVE INVADER ==
3380 FOR I=0 TO 4
3390   IV$(I)=RIGHT$(IV$(I),2)+LEFT$(IV$(I),25)
3400 NEXT
3410 GOSUB 2440'                                 PRINT INVADER
3420 XIV$=RIGHT$(XIV$,2)+LEFT$(XIV$,25)'         LOCATE X
3430 RETURN
3440 '
3450 '== MOVE JUNIOUR INVADER ==
3460 JI$(0)=RIGHT$(JI$(0),26)+LEFT$(JI$(0),1)
3470 JI$(1)=RIGHT$(JI$(1),26)+LEFT$(JI$(1),1)
3480 GOSUB 2520'                                 PRINT JUNIOUR INVADER
3490 XJIV$=RIGHT$(XJIV$,26)+LEFT$(XJIV$,1)'      LOCATE XJUNIOURINVADER
3500 RETURN
3510 '
3520 '== CHECK CRUSH OF JUNIOUR INVADER ==
3530 IF ABLE>0 THEN RETURN
3540 IF MID$(XJIV$,XBEAM,1)="0" THEN RETURN'     MISS !
3550   I=XBEAM'                                  SEARCH FOR LEFT OF 1
3560     IF MID$(XJIV$,I-1,1)="1" THEN I=I-1
3570   MID$(JI$(0),I)="  "'                      DELETE JUNIOUR INVADER
3580   MID$(JI$(1),I)="  "
3590   MID$(XJIV$,I)="00"
3600   GOSUB 3700'                               CRASH JUNIOUR INVADER
3610   GOSUB 3160'                               ERASE BEAM
3620   U=USR2(0)
3630   SCR=SCR+10:GOSUB 4730'                    SCORE COUNT UP
3640   LJIV=LJIV-1:IF LJIV>0 THEN RETURN'
3650     ABLE=5'                                 ATTACK INVADER OK !
3660     SCR=SCR+INT(RND(0)*21+10)*10:GOSUB 4730' 100-300 STEP 10
3670     RETURN
3680 '
3690 '== CRUSH OF JUNIOUR INVADER ==
```

```
3700 COLOR 5
3710   FOR K=0 TO 1'                                    SPARK
3720     LOCATE I,14:PRINT "◢ ";
3730     LOCATE I,15:PRINT " ◤";
3740       FOR J=0 TO 100:NEXT
3750     LOCATE I,14:PRINT " ◣";
3760     LOCATE I,15:PRINT "◥ ";
3770       FOR J=0 TO 100:NEXT
3780   NEXT
3790   LOCATE I,14:PRINT "  ";'                         ERASE
3800   LOCATE I,15:PRINT "  ";
3810 '
3820 '== CHECK CRUSH OF INVADER ==
3830 IF ABLE=0 THEN RETURN'                             IMPOSSIBLE
3840 IF MID$(XIV$,XBEAM,1)<>"2" THEN RETURN'            MISS !
3850   I=XBEAM'                                         SEARCH FOR LEFT OF 1
3860     IF MID$(XIV$,I-1,1)="0" THEN 3870 ELSE I=I-1:GOTO 3860
3870   FOR J=0 TO 4
3880     MID$(IV$(J),I)="      "
3890   NEXT
3900   MID$(XIV$,I)="00000"
3910   GOSUB 4020'                                      CRASH !
3920   GOSUB 3160'                                      ERASE BEAM
3930   U=USR3(0)
3940   SCR=SCR+100:GOSUB 4730'                          SCORE COUNT UP
3950   LIV=LIV-1:IF LIV>0 THEN RETURN'                  EXIST INVADER
3960     SCR=SCR+1000:GOSUB 4730'                       SCORE COUNT UP
3970     SN=SN+1:GOSUB 4780'                            SCENE COUNT UP
3980     GOSUB 2250:GOSUB 2350'                         RESET IV,JIV DATA
3990     RETURN
4000 '
4010 '== CRUSH OF INVADER ==
4020 COLOR 5
4030   LOCATE I, 8:PRINT "     ";
4040   LOCATE I, 9:PRINT " ◢ ◣ ";
4050   LOCATE I,10:PRINT "  ■  ";
4060   LOCATE I,11:PRINT " ◥ ◤ ";
4070   LOCATE I,12:PRINT "     ";
4080     FOR J=0 TO 120:NEXT
4090   LOCATE I, 8:PRINT "  ●  ";
4100   LOCATE I, 9:PRINT "  ●  ";
4110   LOCATE I,10:PRINT "●● ●●";
4120   LOCATE I,11:PRINT "  ●  ";
4130   LOCATE I,12:PRINT "  ●  ";
4140     FOR J=0 TO 120:NEXT
4150   LOCATE I, 8:PRINT "▲  ▲";
4160   LOCATE I, 9:PRINT "▼  ▼";
4170   LOCATE I,10:PRINT "  ◆  ";
4180   LOCATE I,11:PRINT "▲  ▲";
4190   LOCATE I,12:PRINT "▼  ▼";
4200     FOR J=0 TO 120:NEXT
4210   LOCATE I, 8:PRINT "     ";
4220   LOCATE I, 9:PRINT "     ";
4230   LOCATE I,10:PRINT "     ";
4240   LOCATE I,11:PRINT "     ";
4250   LOCATE I,12:PRINT "     ";
4260 RETURN
4270 '
4280 '== CHECK CRUSH OF BEAM GUN ==
4290 I=1
4300 IF I>LMS THEN RETURN'
4310   IF YMS(I)<19 THEN 4480'                          DON'T REACH
4320   IF XMS(I)<XGUN+1 OR XMS(I)>XGUN+2 THEN 4480' SAFE
4330     GOSUB 4520'                                    CRASH BEAM GUN
4340     LOCATE XMS(I),YMS(I)  :PRINT " ";
4350     LOCATE XMS(I),YMS(I)+1:PRINT " ";
4360     GOSUB 2570'                                    PRINT BEAM GUN
4370     U=USR4(0)
4380     FOR J=I TO LMS'                                ERASE MISSILE
4390       XMS(J)=XMS(J+1):YMS(J)=YMS(J+1)
4400     NEXT
4410     LMS=LMS-1
```

```
4420     LBGUN=LBGUN-1:IF LBGUN=0 THEN 4490'     GAME END ?
4430     COLOR 7:LOCATE 4,20:PRINT "LEFT OF BEAM GUN IS";LBGUN;
4440       FOR I=0 TO 4000:NEXT
4450     LOCATE 4,20:PRINT "                             "
4460     GOSUB 2570'                             PRINT BEAM GUN
4470     GOSUB 4830'                             PRINT LEFT OF BEAM GUN
4480 I=I+1:GOTO 4300
4490 GAME=0:RETURN'                              GAME END
4500 '
4510 '== CRUSH OF BEAM GUN ==
4520 COLOR 2
4530   LOCATE XGUN,YGUN  :PRINT "    ";
4540   LOCATE XGUN,YGUN+1:PRINT " ◆◆ ";
4550   LOCATE XGUN,YGUN+2:PRINT "    ";
4560     FOR J=0 TO 100:NEXT
4570   LOCATE XGUN,YGUN  :PRINT " ◆◆ ";
4580   LOCATE XGUN,YGUN+1:PRINT "    ";
4590   LOCATE XGUN,YGUN+2:PRINT " ◆◆ ";
4600     FOR J=0 TO 100:NEXT
4610   LOCATE XGUN,YGUN  :PRINT "◆  ◆";
4620   LOCATE XGUN,YGUN+1:PRINT " ◆◆ ";
4630   LOCATE XGUN,YGUN+2:PRINT "◆  ◆";
4640     FOR J=0 TO 100:NEXT
4650 RETURN
4660 '
4670 '== PRINT HI-SCORE ==
4680   COLOR 7
4690   LOCATE 33,10:PRINT USING "#####";HSCR;
4700 RETURN
4710 '
4720 '== PRINT SCORE ==
4730   COLOR 7
4740   LOCATE 33,13:PRINT USING "#####";SCR;
4750 RETURN
4760 '
4770 '== PRINT SCENE ==
4780   COLOR 7
4790   LOCATE 35,16:PRINT USING "###";SN;
4800 RETURN
4810 '
4820 '== PRINT LEFT OF BEAM GUN ==
4830   COLOR 7
4840   LOCATE 37,19:PRINT USING "#";LBGUN-1;
4850 RETURN
4860 '
4870 '== INSTRUCTION ==
4880 GOSUB 2080'                                PRINTTITLE
4890 RESTORE 5270'                              INVADER
4900   COLOR 7:LOCATE 3,8 :READ I$:PRINT I$;'
4910           LOCATE 3,9 :READ I$:PRINT I$;
4920   COLOR 2:LOCATE 3,10:READ I$:PRINT I$;
4930   COLOR 3:LOCATE 3,11:READ I$:PRINT I$;
4940   COLOR 7:LOCATE 3,12:READ I$:PRINT I$;
4950     COLOR 2:LOCATE 10,9 :PRINT "INVADER";
4960     COLOR 6:LOCATE 10,11:PRINT "  100テン";
4970 COLOR 2:LOCATE 6,14:PRINT "◣◢";'           JUNIOUR INVADER
4980 COLOR 7:LOCATE 6,15:PRINT "▼";
4990   COLOR 3:LOCATE 10,13:PRINT "JUNIOUR";
5000           LOCATE 10,14:PRINT "  INVADER";
5010   COLOR 6:LOCATE 10;15:PRINT "   10テン";
5020     COLOR 5:LOCATE 1,16:PRINT STRING$(37,"-")
5030 COLOR 6:PRINT "  [[ ル - ル ]]"'           RULE
5040   COLOR 4:PRINT "1. ";
5050   COLOR 7:PRINT "スベテ ノ JUNIOUR INVADER ヲ ケサナイト"
5060           PRINT " INVADER ハ コウゲキ デキナイ(シカモ 5 ハツ マデ)。"
5070           PRINT " INVADER ハ マンナカ ヲ ネラウコト !"
5080   COLOR 4:PRINT "2. ";
5090   COLOR 2:PRINT "<< ボーナス >>"
5100   COLOR 7:PRINT "        JUNIOUR INVADER・・・・・・";:COLOR 6:PRINT "100-300 テン"
5110   COLOR 7:PRINT "                INVADER・・・・・・・・・";:COLOR 6:PRINT "1000 テン"
5120     LINE (20,8)-(20,15),"| ",5
5130 COLOR 4:LOCATE 22,9 :PRINT "  << KEY >>";
```

```
5140 COLOR 7:LOCATE 22,11:PRINT "LEFT:4 6:RIGHT";
5150         LOCATE 22,12:PRINT "     SPACE";
5160         LOCATE 22,13:PRINT "    =======";
5170         LOCATE 22,14:PRINT "      BEAM ";
5180   LINE 24,1:COLOR 5:LOCATE 1,24:PRINT"Hit KEY !";
5190   COLOR 3:PRINT " (1:GAME STARET,2:GAME END)";
5200     I$=INPUT$(1):IF I$<>"1" AND I$<>"2" THEN 5200
5210     IF I$="2" THEN STP=1'                        GAME STOP
5220 RETURN
5230 '---------
5240     DATA
5250 '---------
5260 '
5270 DATA " ◢█◣     ":'                               INVADER DATA
5280 DATA "◢███◣    "
5290 DATA "■ ■ ■    "
5300 DATA "◥███◤    "
5310 DATA "_■ ■_    "
5320 '
5330 DATA 01,50,20,C3,2B,C0,21,54,C0,C3,18,C0,21,5B,C0,C3
5340 DATA 18,C0,21,6A,C0,C3,18,C0,7E,A7,C8,46,23,4E,23,CD
5350 DATA 24,C0,18,F4,CD,3B,C0,0D,20,FA,C9,CD,3B,C0,04,0D
5360 DATA 20,F9,C9,CD,3B,C0,05,0D,20,F9,C9,C5,3A,67,EA,CB
5370 DATA EF,D3,40,05,20,FB,C1,C5,3A,67,EA,CB,AF,D3,40,05
5380 DATA 20,FB,C1,C9,0F,50,C8,32,32,64,00,32,32,4B,32,64
5390 DATA 32,96,32,C8,32,64,32,14,32,00,96,1E,32,1E,96,1E
5400 DATA 96,1E,32,1E,96,1E,14,32,00
```

あとがき

あなたもテレビゲームに挑戦

「GAMINGへの招待」

これでおしまいです。いかがでしたか？

“GAMINGへの招待”は，シリーズ1のごあいさつにもありますように

GAMEを題材とした

ソフトテクニックへのアプローチ

であり，その根底には

遊びの精神

がゆったりと流れています。それには，

マシンの違い
言語の違い
レベルの違い

を無視したものでなければいけなく，また

あらゆる分野の遊びのコレクション

が必要だと考えられます。

もともと“GAMINGへの招待”は，シリーズ1(元祖テニス・ゲーム)だけの予定でした。しかしいつかは上記の夢を満たすようなシリーズ群をまとめてみたいと思っています。それは，

ありとあらゆるGAME

のコレクションであり，

ありとあらゆる分野のテクニック

の宝庫です。この

GAMINGへの招待

は，その**試験的試み**だったのです。幸い

第1ブロック

初心者対象のSPACE WAR

第2ブロック

記念すべき“GAMINGへの招待”

シリーズ1の“元祖テニス・ゲーム”

第3ブロック

リアルタイムゲームのBOMBER

第4ブロック

中級者対象のインベーター・パニック

の4シリーズを収めることができました。これを第1ステップとし，**より体系的なもの**をまとめてみたいと思います。読者の御意見をお待ちしています。

御愛読ありがとうございました。

1983年6月1日

MULTIマイコン研究会

塚越一雄

テープ名	内　　　容	定価	機種名	言語	コードナンバー	注文数
月刊マイコン　オリジナル・ソフト						
リアルタイム SUPER STAR TREK	今までのTREKゲームの常識をうち破った傑作。ワープ航法、長距離レーダー始動防御スクリーン作動、積載コンピュータをフル活用して、クリゴンと頭脳戦争だ！	3,000円	PC-8001(32K) FM-7/8 MZ-2000/80B	B	1735 3032 3431	
みみずの滝のぼり	迫りくるゲジゲジの大群に果敢にたち向かうミミズの勇士。でもゲジゲジにつかまると、ゲジゲジが次々と成長し状況悪化。ゆけミミズ戦士よボーナスの日まで！	3,000円	PC-8001(32K)	B	1736	
コードネーム自動表示	ピアノ・ギター楽譜のコード進行チャート、コード修正をスピーディに！ピアノとギターが同時に表示され、またコードを楽譜化して見ることが出来ます。	3,000円	PC-8001(32K)	B	1737	
インデアン・ポーカー	PCとあなたのしのぎを削る賭け金の競い合い。強気になったり、弱気になったり、いかにも人間らしくふるまうPC。あなたとPC、どちらが破産？	3,000円	PC-8001(32K) MZ-2000	M B	1738 3430	
SUPER 卓球ゲーム	本物そっくりの卓球ゲーム。ラケットスイングができ、打球角度を自由にコントロールできます。コンピュータ相手にパーフェクト試合ができたらあなたは天才！	3,000円	PC-8001(32K)	M B	1739	
エイリアンビリヤード	エイリアン相手にビリヤード！あなたのたくみなステックさばきで見事にエイリアンを撃退してください。マシン語&BASICオートスタートです。	3,000円	PC-8001(32K)	M B	1740	
少年とエイリアン	宇宙元年8001年、月面基地に生き残った少年3人と異星人との激しい戦い。勝ち残った少年だけが、栄光のエイリアンレースに参加できます。	3,000円	PC-8001(32K)	M B	1742	
成　績　処　理	①集計表（合計、平均、標準偏差）、②ヒストグラム各種、③素点表（順位、偏差値を含む）、④偏差値表（各教科の偏差値とその散らばり）、⑤個人向けカード、⑥順位表以上の処理が出来ます。1クラス45名で最大7クラスまで可能。	3,000円	PC-8001(32K) FM-7/8	B	1743 3034	
ピラミッドとミイラ	オセロとルービックキューブを組合せた様なゲームで、系統的に考えていかないとなかなか完成しません。たとえ完成出来ずGIVE　UPしてもあとはPC－8001が考えて完成させてくれます。	3,000円	PC-8001(32K)	M B	1790	
ALIEN LAND	人類の平和を守るため、ロボットをうまく操縦して下さい。エイリアンを避けて、エネルギー鉱石を一つでも多く取って下さい。アタックエイリアンに要注意。	3,000円	PC-8001(32K)	M B	1957	
スーパー・ムービング・ブロック	ワーピング・ラケット、攻撃するエイリアン、天じゃステーションが笑っている。ワザ有り、運有り、度胸有り、オールマシン語でスピードも抜群。	3,000円	PC-8001(32K)	M	1958	
ウッドペッカー	緑の木立ちにウッドペッカーがやって来て次々と木を倒してしまいます。さあ、あなたはどれだけウッドペッカーを生け捕りにできるか？	3,000円	PC-8001(32K) FM-7/8	B	1959 3033	
二次方程式解法テクニック	四則計算から正負の計算、文字式、一次二次方程式にいたるまでの解法を、計算の仕方と基本を重視して展開表示します。式を設定するのはあなたです。	3,000円	PC-8001(32K)	B	1961	
SUPERバルーン	ご存知バルルーンボンバーのオールマシン語によるハイスピード版。ビーム砲とバリヤを駆使して飛行機と風船爆弾を迎撃して下さい。	3,000円	PC-8001(32K)	M	1987	
マウンテン・アタック	落石が頻繁に発生、また至るところに人喰い虎が住んでいます。しかも山頂には怪しげな雷雲が……。果して初登頂なるか？	3,000円	PC-8001	B	3784	
CRAZY DESCENDER	クレイジークレイマーばかりがゲームじゃない！　あなたを狙う二人のオジャママンの攻撃をかわしながら無事地上へおりられるか。	3,000円	PC-8001	B	3785	
電気店用 顧客管理システム	1枚のフロッピーには最大630件の顧客が登録可能。顧客の状況をCRT画面上、又はDM用ラベル或は帳表として出力できる。家族状況、商品保有状況、クレジット記録年間月別記録など。	68,000円	PC-8001/FD	ディスク B	3783	
日本語ワードプロセッサー 書記	1ページ最大40字×34行が表示可能！　カナ漢字変換とコード入力、文書はデータとして登録・呼び出しが可能です。印刷と同様に画面表示し、文書の修正・削除も簡単に行なえるなど多彩な文書編集機能を持っています。	9,000円	PC-8801 FM-7/8	B	3617 3021	
SUPERサブマリーン	MICRO-8のグラフィック機能をフルに活用したすばらしいカラー画面です。各種のインジケータを読み取りながら、潜望鏡をのぞいて、敵戦艦20隻を魚雷で撃沈させて下さい。	3,000円	FM-7/8	B	3030	
THE BASEBALL	投手はスロー、スピードボールを選択して投球します。またチェンジアップも可能です。また打球もフライ、テキサス性のヒット、ゴロなど、さまざまです。FM8のすばらしいカラーグラフィック画面で楽しんで下さい。	3,000円	FM-7/8	B	3031	
スペースタッチダウン	ナインドは1,000～300まで、4つの画面に着陸基地・宇宙船・燃料・加速度が表示されています。宇宙船を操縦して無事着陸して下さい。	3,000円	FM-7/8	B	3035	
実戦プロ野球ゲーム	ダブルプレーが実現!!セントラルリーグをFMシリーズでどうぞ。	3,800円	FM-7/8	B	3022	
実戦グラフィック麻雀ゲーム	ポン・チー・カン、あの感触をマイコンで……。	3,800円	FM-7	M B	3951	
花札（コイコイ）	さあ来い！勝負だ!?花札の醍醐味を楽しんで。	3,800円	FM-7 X-1	M B	3950 3281	
ヨットシミュレーション	海図と風向、コンパスをたよりにゴールをめざせ。風をとらえてうまくタッキングして下さい。史上初の航海シミュレーションゲーム。	3,800円	PC-8800 FM-7/8 X-1	B	3631 3978 3280	
トランプ・パッケージ	セブン・ブリッジ、ポーカー、ブラック・ジャックの3タイトル入った徳用パッケージ。最高のグラフィックでトランプを楽しんで下さい。	3,800円	FM-7/8	B	3979	
パッチ・パズル	CRTでジグソーパズルをどうど。FMの頭脳をあなたは超えられるかな。美しいグラフィック・パズルです。	3,000円	FM-7/8	B	3985	
オイチョ・カブゲーム	日本古来のカードゲーム。胴元はもちろんマイコンです。よく考えて勝負して下さい。熱くなって身ぐるみはがされないように。	3,000円	MZ-80B	B	1744	
THE　ギャング	大金が眠る豪邸の金庫へたどりつくには、数々の迷路のワープトンネルを利用して突破しなければならない。超一流のギャングである君の行く手を待つのは、大金持ちへの道か、冷たい牢獄か、はたまた大爆発か！	3,000円	PA-7010 パソピア	T B	1960	
日本対JR	JR軍団が攻めてきた。勇敢な君は零戦とジェット機を操縦して宿敵JR軍団をやっつけて下さい。	2,800円	JR-100	B	1066	
THE GOLF	あちこちに点在する池と林を計算して、方向と飛距離を決定して下さい。池に落ちたり林に当ったり、コースからはずれると、打数が加算されます。うまくグリーンをとらえられるかな？	2,800円	PC-6001	B	1048	
三次元カーレース	スリル満点。追いつ、追われつ立体カーレース。	3,500円 3,200円	MZ-700 MZ-1200	M M	3511 3466	

言語　B：ベーシック，M：機械語，HU：HuBASIC，F：FORM-B

テープ名	内容	定価	機種名	言語	コードナンバー	注文数
クラッシュ クリーン	2匹のエイリアンをさけながらあなたの住む町をきれいにしてゆく(白く塗ってゆく)ものです。ワープトンネルを利用して、何面までクリーンに出来るかな？でもエイリアンもワープするから御用心。	2,800円	MZ-1200	B	1897	
三次元ゴルフ	MZ-700で初の本格的シュミレーションゴルフです。飛距離と方向はあなた次第です。	3,500円	MZ-700	M	3559	
リークの大逆襲	ロボットは3つ。アミダでパターンが決まります。○を全てコの字形の中に納めるか、またはリークが○にぶつかると点になります。	2,800円 3,000円	MZ-1200 PC-8001	B B	3467 3819	
金鉱掘りゲーム	地中海には莫大な金がねむっています。金脈を見つけて一かく千金をねらって下さい。	2,800円 3,000円	MZ-1200 PC-8001	B B	3468 3820	
PC-8801用 実戦プロ野球ゲーム	野球好きのあなた、一度野球チームの監督になってみませんか！　もちろんチームは巨人・阪神。江川・中畑・原・篠塚、掛布・岡田が打って走る。	3,800円	PC-8801	B	3619	
THE GOLF	本格的なゴルフシュミレーションゲーム。あちこちに点在する池と林を計算して方向と飛距離を決定。うまくグリーンをとらえられるかな？	3,000円	PC-8001	B	3821	
グラフィックシュミレーション・スターウォー	ブラックホール・戦闘・戦略などアクションがいっぱい。大宇宙戦での指揮官はあなたです。	3,800円	PC-8801	M	3624	
ボイジャー・アタック	ベータ恒星系・第3惑星の中立地帯でクリンゴン帝国の宇宙編隊と遭遇。	3,800円	PC-8801	M B	3626	
シーウォー	海の怪物が襲って来る。あなたは深海底(3面以上)の戦いにたえられるか？	3,800円	PC-8801	M B	3625	
実務トレーニング80B・2000用						
価値判断	マイコンなら入力されたデータにより色付けなしの価値判断が可能。	3,000円	MZ-80B	B	1701	
ローン計算	世はまさにローン一色。マイコンに算出させるのがナウい方法。	2,800円	MZ-80B	B	1704	
多角形の面積計算	もっともポピュラーな多角点測量のデータを計算しデータを求めるプログラム。	3,000円	MZ-80B	B	1705	
多元連立方程式	二元以上、27元までの連立一次方程式を消去法で解答します。	2,800円	MZ-80B	B	1706	
ニュートン法	方程式f(X)=0の近似値解を求めるために、微分を使って算出するソフトウェア。	2,800円	MZ-80B	B	1708	
QSO整理	QSO(交信)記録を手書で整理する時代は、このソフトの登場で終わった！	3,500円	MZ-80B	M	1784	
成績処理(4本組)	1ページ最大20項目×50人×10ページの格納が可能！5段階変換、偏差値変換、生徒番号順一覧、成績順一覧、ヒストグラム、クラス別一覧、クラス別成績順一覧、全クラス総合一覧、個人表など19種類の表が作成可能です。	24,000円	MZ-80B	B	3335	
S-P表作成(2本組)	S－P原表(1.0得点、重みつき誤答分析)、注意係数、平均正答率、信頼性係数、差異係数、標準偏差、S－P表、項目(問題分析)、UL指数、φ係数、クロス表、ソマーズ係数、ヒストグラム、部分S－P表、多肢選択式誤答分析、ヒストグラム、クロス表、集中度係数、平均情報量、相対エントロピピー実質選択肢数、偶然得点など。	12,000円	MZ-80B	B	3336	
アンケート集計	1ページ最大250人×10ページのDATA入力が可能、項目(問題)の選択肢数の最大は35、集計結果一覧表、ヒストグラムクロス分析表、クロス分析ヒストグラムがプリントされます。	6,000円	MZ-80B	B	3337	
校内模試(3本組)	クラス数は最大11クラス、1クラス50人です。各科目別ヒストグラム、各科目別得点、偏差値、学年順位一覧、各科目別成績順位一覧表、各クラス別偏差値一覧、クラス別成績順位一覧、学年総合成績順、全クラスの個人成績表出力など。	18,000円	MZ-80B	B	3338	
簡易言語．HU-BASIC．開発ツール						
〈MATRIX-AUTO-REPORTING-SYSTEM〉 MARSマーズ 表集計/グラフ・パッケージ	行列の四則演算／自動プリントアウト／レイアウト指定プリントアウト／行・列のページ別管理(3次元処理・DISK版)／グラフ作成／X－Yプロッターによるグラフ作成／行・列のレイアウト変更／条件判断／ソーティング／計算式の指定	9,800円 14,800円 14,800円 24,800円 29,800円	PC-8001/T PC-8001/FD PC-8801/T PC-8801/FD PC-9801/FD	B	3857 3856 3635 3634 3732	
HU-BASIC V2.0	200種類を越える豊富な命令群。Version1.シリーズをはるかにしのぐ高速性。このSUPER-BASICで今MZが甦る。	10,000円	MZ-2000	B	3389	
HU-BASIC V2.0	200種類を超える豊富な命令群。Version1.シリーズをはるかにしのぐ高速性。このSUPER-BASICでMZが今甦る。	10,000円	MZ-80B	B	3390	
HU-BASICV2.0/FD	MZユーザー待望のHU-BASICのフロッピー・バージョン。スピーディなローディングでSUPER-BASICを思うままに。	20,000円	MZ-80B/FD MZ-2000/FD	B	3397 3398	
HUBASICコンパイラー	整数型BASIC→マシン語変換プログラム。ゲームソフトの作成を目的として開発された簡易言語です。BASICと同様の文法に従ってプログラムを作成し、テストRUNにより文法チェックが可能です。その後コンパイラーによりプログラムをマシン語変換をします。	各6,000円	X-1 PC-8001MKII MZ-700	M	3182 3875 3483	
HU－CAL	最高級ビジネス用簡易言語です。 ◎16桁の高精度計算　◎倍精度関数を装備　◎フルスクリーンエディターによる編集機能付　◎フォーマット指定が容易　◎フィールド調整はカーソルキーでOK　◎MZ-700用はプロッター使用可。	9,800円 9,800円 8,800円 8,800円	MZ-2000 MZ-80B MZ-700 MZ-1200	M M M M	3393 3394 3395 3396	
HU-CAL／FD	HU-CALのフロッピー版です。スピーディな操作性でさらにカセットバージョンHU-CALのデータがそのまま使用できます。	19,800円 19,800円	MZ-2000/FD MZ-80B/FD	M M	3391 3392	
Pocket INFO	収集したデータを、表集計の形式でプリンターへ打ち出し、指定した条件でデータの検索、ソートが可能です。またデータをカセットへSAVE, LOADができ、さらにMZ-2000とのデータの送受信が出来ます。	6,800円	PC-1500 ㊖E-155	B	3160	
Super INFO	充実したヘルプ画面が認けられているので簡単にプログラミングが可能な簡易言語です。Pocket INFOとのデータの送，受信が出来ます。	29,800円	MZ-2000/FD	B	3439	
HU-BASIC COMPILER	高能力言語HU-BASIC用の32Kバイトマシン語コンパイラ。120行分のコンパイル可能。最敵化機能により実効時が最小ですむ(マニュアル付)。	10,000円	MZ-80K/C	M	1785	

ゲーム・ソフト for PC-8001mkII

テープ名	内容	定価	コードナンバー	言語	備考	注文数
スーパーゴルフ	mkII のグラフィックを生かしたリアル・ゴルフ	3,800円	3833	BM	(32K)	
夢のプロ野球	巨人VS西武、夢のカードが実現。	3,200円	3834	B		
スーパーグラフィックマージャン	MKIIのグラフィックを生かした高速マージャン。	3,000円	3837	M		
爆弾男	時限爆弾を仕掛けてそれ逃げろ、君は必殺爆弾仕掛人。	3,200円	3848	M		
ガンマン	君は西部のガンマン。弓を射ってくるインデアンをやっつけろ。	3,200円	3849	M		
ミスターバラフライ	撃っても撃っても卵→青虫→サナギ→蝶と変態。もうタイヘン。	3,800円	3850	M		
ベジタブルクラッシュ	フォークを武器に、野菜君たちと戦いだ。	3,800円	3851	M		
キャノンボール	はねるボールをうまくよけながら割って下さい。	3,800円	3852	M		
あなたは名カメラマン	突然現われる森の動物たちの瞬間をキミはとらえられるかな。	3,200円	3853	M		
DATA BASE	住所、レコード、QSOカード等、すべての情報ファイルはこれで	3,800円	3854	M		
ビンゴ25	たて、よこ、ななめ。早く5列に並べれば勝。	2,800円	3855	B		
ひつじや～い	牧歌メルヘン調ゲームです。	3,800円	3859	M	NEW	
MJ－05	地球の運命を握るのはあなたです。	3,800円	3860	M	NEW	
HELP	ヘビとあなたの知恵くらべ。	3,800円	3861	M	NEW	
華麗な怪盗	カジノの金をかっぱらえ。	3,800円	3862	M	NEW	
Justice Knight	マイコンはメルヘン。そしてファンタジー。	3,800円	3863	M	NEW	
フィールドウォーズ	地雷をセットしてオバケをやっつけろ。	3,800円	3864	M	NEW	
スーパードアーズ	ドアの反動を利用してエイリアンをやっつけろ。	3,800円	3865	M	NEW	
バブルクンド1999	伝説の都バブルクンドを守るのは君だ。	3,800円	3866	M	NEW	
パワーフェイル	CPU開発センターを守れ。	3,800円	3868	M	NEW	
明るい農園	壮大な自然を相手にくりひろげるゲーム。	3,800円	3869	M	NEW	
ピラミッドアドベンチャー	ピラミッドの財宝を探せ！	3,800円	3870	M	NEW	
カエル・シューター	怪物カエルが襲って来るゾ！	3,800円	3871	M	NEW	
ギョウザ・パニック	雨あられと落ちて来る餃子を受け止めろ！	3,200円	3872	M	NEW	
ローリング・シューター	飛来するエイリアンをやっつけろ。	3,200円	3873	M	NEW	
ブーメラン・ハンティング	必殺ブーメランで獲物をとりましょう。	3,800円	3874	M	NEW	
ひよこファイター	ヒヨコになってタマゴを食べよう。	3,200円	3876	M	NEW	
サブマリンシューター	機雷を攻撃しながら深く潜航せよ。	3,800円	3877	M	NEW	
スカイダイバー	パラシュートをつかってうまく降下せよ。	3,200円	3878	M	NEW	
イタサンドリアス	地底タンクに乗ってヘクトリアンをやっつけろ。	3,600円	3879	M	NEW	

ゲーム・ソフト for PC-8001

テープ名	内容	定価	コードナンバー	言語	備考	注文数
四人麻雀	PCでマージャンはいかが	3,000円	3838	M		
スペースドッチャー	左右から攻撃して来るUFOを撃墜しよう。	3,600円	3832	BM		
シリウスF	君は地球へ帰ることができるか。	3,000円	1916	BM	(32K)	
エアーライフル	11発の弾丸で君は何点かせげるかな？	3,000円	1917	BM	(32K)	
キングタイガーIII PART1	"キングタイガー"迎え撃つシャーマン戦車。	3,000円	1918	BM	(32K)	
バトルバルカン	君は宇宙パトロールの隊員だ。	3,000円	1919	BM	(32K)	
ビッグアステロイド	無事に地球へ帰還ができるか。	3,000円	1920	BM	(32K)	
ブラックホール	新兵器プロトン砲を使いホワイトホールへ脱出しろ。	3,000円	1921	BM	(32K)	
戦艦大和	特命を受けた戦艦大和は沖縄に向け出発する。	3,000円	1922	BM	(32K)	
ドキドキすいか割り	すいか割りを楽しんでいる。ところが…。	3,000円	1923	B	(32K)	
アスロック	海上には戦艦、機雷が雨のように降ってくる。	3,000円	1924	BM	(32K)	
ワイルドスワット	暴走族絶滅の為、今日もパトロール。	3,000円	1925	BM	(32K)	
プラネットバルカン	宇宙歴2999年。遂に惑星間戦争に突入してしまった。	3,000円	1926	BM	(32K)	
ギャラクティカ 1	スクリューカノン砲で敵を破壊せよ。	3,000円	1927	BM	(32K)	
ギャラクティカ 2	反乱軍は侵入者に対して無差別攻撃を加えてくる。	3,000円	1928	BM	(32K)	
ギャラクティカ 3	太陽の重力変化の為絶滅の危機。	3,000円	1929	BM	(32K)	
バトルファイヤー	せまりくる敵大船団。	3,000円	1930	BM	(32K)	
CRTチェイサー PART1	10ヶのチェックポイントを周らねばなりません。	3,000円	1931	BM	(32K)	
CRTチェイサー PART2	宇宙機雷と目に見えない境界線に注意。	3,000円	1932	BM	(32K)	
CRTチェイサー PART3	探査装置をかついで、埋蔵金を探しあてて下さい。	3,000円	1933	BM	(32K)	
CRTチェイサー PART4	あなたは地底の迷路に迷いこんでしまいました。	3,000円	1934	BM	(32K)	
マリンどんべえだあ PART1	汚染の海を清掃中。回収のゴミの中に不発弾も交っている。	3,000円	1935	BM	(32K)	
マリンどんべえだあ PART2	どんべえII世号は外洋掃除の任務につきました。	3,000円	1936	BM	(32K)	
スカイどんべえだあ PART1	大気汚染調査隊は今日も調査に出かける。	3,000円	1937	BM	(32K)	
スカイどんべえだあ PART2	2機で回ることになったスカイどんべえ。	3,000円	1938	BM	(32K)	
スペースどんべえだあ PART1	エイリアンは惑星の陰にかくれています。	3,000円	1939	BM	(32K)	
スペースどんべえだあ PART2	奴らのミサイルはなかなか強力です。	3,000円	1940	BM	(32K)	
スペースランディング	宇宙ステーションマザーIに着艦しなければならない。	3,000円	1942	BM	(32K)	
フューチャー	ブラックホールに吸いこまれたら生きて帰れない。	3,000円	1943	BM	(32K)	
侵略ゲーム	直射、斜撃砲を使って撃ち抜き防戦。	3,000円	1989	B	(32K)	
大脱走ゲーム	ランクも3段階と充実。	3,000円	1990	B	(32K)	
七並べトランプ	トランプゲームコンピュータ相手に楽しもう。	3,000円	1993	B	(32K)	
迷探偵ゲーム	通り過ぎる犯人をつかまえよう。	3,000円	1994	B	(32K)	
ファイアー・インフェルノ	ビルが大火事。さあたいへん。	3,000円	1995	B	(32K)	
スクランブルチェイサー	追いつ追われつ、2人で楽しみ3倍増。	3,000円	3763	BM	(32K)	

ゲーム・ソフト for PC-8801

テープ名	内容	定価	コードナンバー	言語	備考	注文数
ザ・ゴルフ	ハラハラ・ドキドキ！本格的シミュレーション・ゴルフをどうぞ。	3,000円	3620	B		
PP－2	フライトシュミレーターとポラリスの2本パッケージ・ブック型式マニュアル付。	4,000円	3621	M		
スペース・ビー	ハチの巣を狙え！ギャラクシアン・ビー	3,000円	3622	M		
ミサイル・ディフェンダー	敵機を探せ！ミサイル発射！	3,000円	3623	M		
ラブアントラブ	女王様救出ゲーム。リンゴやケーキを運び出せ!!	3,000円	3627	M		
4人マージャン	漢字ROM不要。人が出て来るよ。	4,800円	3628	B		
シミュレーション ミッドナイト・コマンダー	君は勝てるか！真夜中の海戦。	3,800円	3633	B	NEW	
キックオフ	ラグビーボールをエイリアンに向って蹴り込め。	3,000円	3636	M	NEW	

お申し込みは、現金書留か郵便振替で、右記送料をそえて、電波新聞社へ送金下さい。

送料　1本…………200円　2本以上1本増すごとに100円　5本以上は送料無料

テープ名	内容	定価	コードナンバー	言語	備考	注文数
レーダースネーキー	レーダーをのぞきながらニョロリ、ニョロリ。	3,000円	3637	M	NEW	
FG8801	サブマリン他4タイトルゲームパッケージ。	8,800円	3638	M	5インチFD	

PASOPIA7

テープ名	内容	定価	コードナンバー	言語	備考	注文数
ベジタブル・クラッシュ	フォークを武器に野菜君たちと戦いだ！	3,800円	1274	M	NEW	
キャノンボール	はねるボールをうまくよけながら割って下さい。	3,800円	1275	M	NEW	
爆弾男	時限爆弾を仕掛けてそれ逃げろ!!君はプロの必殺爆弾仕掛人。	3,800円	1276	M	NEW	
ザ・スパイダー	飛来する宇宙グモを攻撃せよ。	3,800円	1277	M	NEW	
カエルシューター	目前に立ちはだかる蛙。スリルとスピード感あふれる三次元立体ゲーム。	3,800円	1278	M	NEW	
ヒヨコファイター	ヒヨコになってタマゴを食べよう。	3,200円	1279	M	NEW	

ゲーム・実務トレーニング・ソフト forパソピアPA-7010

テープ名	内容	定価	コードナンバー	言語	備考	注文数
マーズ・パニック	侵入して来たモンスターを落し穴へさそい込んで埋めて下さい。	3,800円	1247	BM		
ゴルフゲームスペシャル	茶の間で居ながらにしてゴルフを楽しんで下さい。	3,800円	1249	BM		
麻雀	メンツがいなくても、マイコンで4人マージャンができる。	3,800円	1273	BM	R1.1専用	
金種計算	経理課の悩みの種もこれで楽々！	2,800円	1201	B		
アルデバラン #1	スタートレックをしのぐBASICゲームの	3,600円	1202	B		
スーパースタートレック	マイコンゲームの古典的名作。	3,800円	1203	B		
ビンゴ 25	たて、よこ、ななめ、早く5列に並べた方が勝ち。	3,600円	1204	B		
アニマルレッスン	マイコンは動物の知識を増やそうと必死	3,000円	1205	B		
頭の体操 No.1	四つのジャンルをテストします。	3,000円	1206	B		
頭の体操 No.2	頭の体操No.1を中級クラスまでアップ。	3,200円	1207	B		
頭の体操 No.3	これで高得点が取れたら尊敬します。	3,400円	1208	B		
キーボードレッスン	正確なインプットをするためのソフトウェア。	3,200円	1209	B		
ハングマン	マイコンが指定するシークレット・ワード。	2,800円	1210	B		
殿様ゲーム	あなたはエゾの国の大将。	2,800円	1211	B		
株式売買ゲーム	5銘柄の相場を50日間取引する。	2,800円	1212	B		
チェッカー	相手のコマを飛びこすとコマが取れる"チェッカー・ゲーム"	3,000円	1213	B		
英会話レッスン（上級）	英会話でよく使う表現の基本編。	3,400円	1214	B		
英会話レッスン（中級）	英会話でよく使う表現の応用編。	3,200円	1215	B		
ベーシック・レッスン No.1	コマンドの説明用プログラム。（入門編）	3,000円	1216	B		
ベーシック・レッスン No.2	コマンドの説明用プログラム。（基礎編）	3,000円	1217	B		
ベーシック・レッスン No.3	コマンドの説明用プログラム。（応用編）	3,000円	1218	B		
ニュートン法	方程式f(x)＝0の方似値解を求める。	2,800円	1219	B		
多元連立方程式	二次以上、27元までの連立一次方程式を解答。	2,800円	1220	B		
月面着陸	距離と高度を見ながら着陸。	3,600円	1221	B		
バリケード	星を壁にぶつからないように。	3,000円	1223	B		
ブロックくずし	ボールは全部で7個。さて何点とることができるか？	3,200円	1224	B		
陣取りゲーム	相手に陣地をとられないように。	3,000円	1225	B		
ハノイの塔	並んだ円盤を崩さず移動しましょう。	3,000円	1227	B		
占星術	相性、恋愛運等も教えてくれる。	4,200円	1230	B		
医は算術なり	医学生必携のゲーム？	3,400円	1231	B		
価値判断	コンピューター的使い方決定版！	3,000円	1232	B		
英単語レッスン（初級）	設問にキーボードで解答。	2,800円	1233	B		
英単語レッスン（中1用）	設問にキーボードで解答。	3,000円	1234	B		
英単語レッスン（中2用）	設問にキーボードで解答。	3,000円	1235	B		
英単語レッスン（中3用）	設問にキーボードで解答。	3,000円	1236	B		
モールスレッスン	パソピアでモールス練習。	3,000円	1238	B		
ローン計算	簡単なデータ入力でマイコンに算出。	3,000円	1239	B		
測量計算	三点の座標を入れて面積を計算する。	2,800円	1240	B		
多角形の面積計算	多角点測量のデータを計算。	2,800円	1241	B		
2001年宇宙の旅 PART1	HAL 9000の反乱を、どう止めるか！	3,500円	1242	B		
2001年宇宙の旅 PART2	スタートゲートを通り過ぎた。そこにはコクセキヒが……。	3,500円	1243	B		
ウォーク・ワン	13丁目の聖子ちゃんの家に行くが……。	3,500円	1244	B		
コンピューター BuG9000	HAL-9000を作ろうとしていたが……。	3,500円	1245	B		
ザ・ゴルフ	フルカラー・ゴルフをお楽しみ下さい。	3,000円	1246	B		

ゲーム・ソフト for PC-6001

テープ名	内容	定価	コードナンバー	言語	備考	注文数
ビンゴゲーム	コンピューター相手にマス目つぶし。	3,200円	1000	BM		
アルデバラン #1	SFストーリー・ゲームのPC-6001版。	3,200円	1001	BM		
チェッカー	相手のコマを飛びこすとコマが取れるチェッカーゲーム。	3,000円	1003	BM		
株式売買ゲーム	相場師としてのウデマエをためすいいチャンス！	3,000円	1006	BM		
スペースシューティング	UFOをレーダースクリーンでとらえて撃破。ジョイスチック可。	3,400円	1007	BM		
医は算術なり	ハタで見るほどお医者さんも楽ではないですね。	3,000円	1009	BM		
大戦車突破作戦	マシン語によるタンクゲーム。ジョイスティック可。	3,800円	1010	BM		
バルチック艦隊	バルチック艦隊接近！砲台を死守せよ。ジョイスティック可。	3,200円	1042	BM		
アステロイド エクスプレス	飛び交う小惑星をかいくぐり、どこまで行けるかな。ジョイスティック可。	3,400円	1043	BM		
プロレーサー	ゲームセンターでおなじみのカーレーサー。ジョイスティック可。	3,200円	1044	BM		
タイタンファイター	タイタンの侵入者から地球を守れ！地球の運命は？	3,200円	1045	BM		
スペースビー	宇宙バチが襲ってきた、ハチの巣を狙え!!	3,000円	1047	B		
FX空中戦	バリヤーの中で何機コアファイターを破壊できるか。ジョイスティック可。	3,200円	1046	BM		
ブロックくずし	ブロックくずしの超高速版。パドルの大きさと、球速は調整可。	3,000円	1025	BM		
U・F・Oくずし	ブロックに囲まれたUFOを撃墜して下さい。	3,000円	1026	BM		
五目ならべ	パピコン相手に五目ならべはいかが。	3,000円	1027	BM		
ミステリーハウス	アドベンチャーゲームの決定版。	3,800円	1024	B		
PC-6001用 百人一首	学習とゲームを両立したソフトテープ登場。	3,000円	1049	B	(32K)	
ダイヤモンド アドベンチャー	ビルに隠されたダイヤをデジタルロックのNo.を探して発見しよう！	3,500円	3101	B	(32K)	
クラッシュ・ラリー	何個のポイントマークをとることができるか。	2,800円	3102	BM		
ブラックボックス	ボールをレーザービームでうまく発見できるか。	2,500円	3104	BM		
UFOゲーム	敵円盤は高性能ランダム走行メカを揃えている。	2,800円	3105	BM		
ギャラクティカ ①	ゴモラがミサイル攻撃をしかけてきた。	2,800円	3107	BM		
バグ・パニック	奇怪な虫が出没。強暴な土人が襲ってくる。	2,800円	3111	BM		
エイリアン	君の武器は如意棒と、ずばぬけたジンプ力だけ。	2,800円	3112	BM	(32K)	

月刊 マイコン別冊 **PC-8001 PC-8801 マシン語入門**

マシン語の基礎から ゲームの製作まで

■あなたは自分のマシンを使いこなしているか？　現在BASICを理解できる人は、マイコン所有者の10分の1、マシン語に至っては、そのまた何分の1かである。本書は〝マシン語修得のための近道〟として作られた。

〒300円

B5判　210頁　定価1,300円

月刊 マイコン別冊 **PC-8001マシン語入門II**

アセンブラから電子音楽付き カラー・グラフィックまで

■好評のPC−8001、8801マシン語入門に続く第二巻。例題を通し実際に自分の目で確めながら学習を進めている。今回は、アセンブラから効果音付きカラーグラフィックにまで挑戦。

〒250円

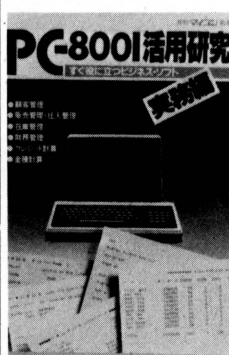
B5判　218頁　定価1,300円

月刊 マイコン別冊 **PC-8001活用研究（ゲーム編）**

■PC−8001の機能などを細かく、初心者の方にもわかり易く説明。プログラムも実務的なものから、ゲームまで広範囲にわたって利用でき、またプログラムの選び方、特徴なども紹介されている。

〒250円

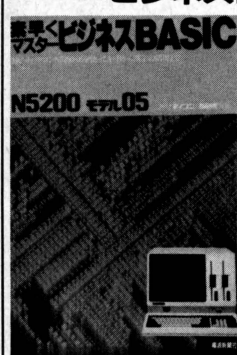
B5判　198頁　定価1,200円

月刊 マイコン別冊 **PC-8001活用研究・実務編**

すぐに役に立つ ビジネス・ソフト

■NECのPC−8001用ビジネスソフトを6本紹介。プログラムリストとフローチャートの二つを掲載、変数の使い方が難しいものについては変数表をのせた。

〒300円

B5判　154頁　定価1,500円

月刊 マイコン別冊 **素早くマスター ビジネスBASIC**

N5200 モデル05

■パソコンのビジネス活用が進むなかで、NECの最上位16ビットパソコン「N5200モデル05」を使い、イラストを多用し、見ながら自然に理解できるようやさしく解説したビジネスBASICの入門書。

〒250円

B5判　186頁　定価1,500円

月刊 マイコン別冊 **MZ-2000活用研究**

テクニックから機能・各種アプリケーションまで

■機械の特徴、性能を徹底的に分析、MZ−80Bとの互換性をわかり易く紹介。汎用プログラムとしてゲームプログラム、日本語ワードプロセッサの活用、BASICコンバートプログラム、実務プログラムを各種紹介。

〒250円

B5判　178頁　定価1,300円

月刊 マイコン別冊 **Z-80マイコン プログラムテクニック**

■ザイログ社の開発したCPUZ80が、手軽さと汎用性のため、8ビットマイクロコンピュータの主流になっている。本書はZ80の命令をわかりやすく解説。またテストプログラム，演算プログラムも各種紹介している。

〒250円

B5判　238頁　定価1,300円

月刊 マイコン別冊 **MZ-80B活用研究**

実践プログラム集

■シャープ㈱のクリーンコンピュータMZ−80Bの機能を徹底分析プログラムの遊び方、特徴など紹介し、最後にプログラムリストを掲載。また、巻末にはBASIC……HuG BASICの解説をしている。

〒250円

B5判　234頁　定価1,300円

伝票

帖合	書店名

注文誌名	発行所
	電波新聞社

定価　　　　円

部数　　　　部

注文　　月　　日

注文書	
住所	様
氏名	
電話	

本屋さんへのお申し込みは、この注文伝票をお渡し下さい

マイコン別冊
GAMINGへの招待
塚越一雄 著

昭和58年8月30日発行

定価 1,300円（送料250円）

発 行 人　平山秀雄
発 行 所　(株)電波新聞社
郵便番号141 東京都品川区東五反田1—11—15
電話(03) 445—6111(大代) 振替東京5—51961
印 刷 所　大日本印刷(株)
製 本 所　(株)堅 省 堂

電波新聞社　〒141東京都品川区東五反田1－11－15　電話03(445)6111　定価1,300円　雑誌68469-0